Dynamic Quantification in Logic and Computational Semantics

Ian Lewin

Ph.D.
University of Edinburgh
1992

Abstract

Dynamic Interpretation can be characterized by the idea that when we understand some part of language, we not only use contextual features to help us in interpreting it but we also, as a result of interpreting it, generate a new context in which further interpretation can take place. The output context for one expression can become the input context for another. Contexts are thereby *threaded* through the interpretation of language. This thesis extends current ideas in Dynamic Semantics in two directions: first, by defining a precise dynamic truth definition for a fragment of English, rather than a strictly logical language; and secondly, by defining a dynamic logic with binary structured quantifiers where information from the first argument is threaded into the second.

Declaration

I composed this thesis myself in its entirety. It describes my own research.

Ian Lewin

Edinburgh

July 23, 1992

Acknowledgments

First, I thank my supervisors Robin Cooper and Graeme Ritchie for all their work on my behalf. They have listened to, read and criticised my ideas from the inchoate to the implemented. I also thank my colleagues in the Department of Artificial Intelligence and the Centre for Cognitive Science at Edinburgh University for the academic milieu but above all for their friendship. Perhaps I might mention in particular Alan Black, Flávio Corrêa da Silva, Matthew Crocker, Martin Emms, Carla Gomes, Nelson Ludlow, Suresh Manandhar, Brian Ross and Rob Scott.

It was the First European Summer School for Logic, Language and Information held at the University of Groningen in 1989 that introduced me to dynamic semantics (or 'demonic psymantics' as Alan Black once accidently and perhaps aptly called it). For the financial assistance to attend that school and, moreover, for a three year grant enabling me to undertake higher education (SERC research studentship 88304590) I acknowledge the support of the Science and Engineering Research Council and thank their sponsors, the British Government and their sponsors modern British society.

Finally, I shall thank my mother who believes, no doubt correctly, that it was her idea for me to return to university to study for a higher degree.

For my mother and father

'Were there no advantage to be reaped from these studies, beyond the gratification of an innocent curiosity, yet ought not even this to be despised; as being one accession to those few safe and harmless pleasures, which are bestowed on human race. The sweetest and most inoffensive path of life leads through the avenues of science and learning'

David Hume: An Enquiry concerning Human Understanding, 1748.

Contents

Chapter 1

Introduction

The interpretation of language and the linguistic contexts in which it takes place depend on each other. When we understand a part of language, we not only use contextual features to help us in interpreting it but we also, as a result of interpreting it, generate a new context in which further interpretation can take place. Dynamic semantics formalizes both a notion of interpretation *in* a context and a notion of context generation *by* interpretation. The interpretation of an expression is made relative to a pair of contexts, which can be called respectively the input context and the output context. The output context for one expression can become the input context for another. Contexts are *threaded* through the interpretation of language. In particular, the system of Dynamic Predicate Logic [Groenendijk & Stokhof 91b] has been designed to capture the same data concerning possible anaphoric relations in English as the original version of Discourse Representation Theory [Kamp 81]. For example, the output context for an existentially quantified formula records a particular value for the variable quantified over. When this output context is threaded as input into the next formula, a subsequent similar variable can take the same value even though it is not in the usual semantic scope of the existential quantifier. In this way, certain relations between quantified noun phrases and dependent pronouns can be captured in a way which was not possible before in the tradition of Montague Grammar [Montague 74a].

The work described in this thesis examines and extends current work in dynamic se-

mantics in two main directions. First, we define a dynamic semantics for a fragment of English with simple surface syntactic structures. The system of Dynamic Predicate Logic (DPL) only defines a semantics for the syntax of first order logic and, as a result, the language interpreted dynamically is already disambiguated for relative quantifier scoping behaviour and pronominal resolution. English syntax, however, does not appear fully to determine such disambiguations. That is, it is plausible to say that the sentence 1a has just one syntactic structure but two possible meanings paraphrased by 1b and 1c.

(1) a Every owner bought a tin
b For each owner there is a tin he bought
c There is at least one tin which every owner bought

Similarly, the second sentence of 2

(2) The policeman hailed the priest. He greeted him

has just one syntactic structure but could be true either if the policeman greeted the priest or the priest greeted the policeman. The two issues of quantifier scoping and pronominal resolution are also *jointly* important because there are examples where the possibility of certain anaphora depends on relative quantifier scoping behaviour. For example, there is no interpretation of 3 where 'he' is anaphorically related to 'a doctor' and where 'every patient' takes wider scope than 'a doctor'.

(3) A doctor saw every patient and he was completely exhausted afterwards

If English syntax does not determine these disambiguations then we cannot expect to give a dynamic semantics for English by first using purely syntactical considerations to translate into DPL and then invoking the semantics of DPL. In this thesis, a relation between dynamic logic and actual English is precisely stated and used in the analysis of sentences in which relative quantifier scoping and anaphoric possibilities interact.

The second contribution of the thesis lies in extending the expressive power of DPL. Whereas Dynamic Predicate Logic only contains quantifiers which take one argument,

we define a dynamic semantics for a logic containing quantifiers that take two arguments. The investigation of binary structured quantifiers has been a highlight of semantical investigations since the publication of [Barwise & Cooper 82]. In an English quantified sentence such as

(4) most owners that expressed an opinion said that their cats preferred it

the determiner 'most' is taken to denote the quantifier and the complex noun attached to the determiner – 'owners that expressed an opinion' – denotes the first argument to the quantifier. The second argument is denoted by the verb-phrase 'said that their cats preferred it'. These types of structure have been independently important on account of the possibility of an anaphoric relation between a noun phrase in the first argument and a pronoun in the second. These structures are traditionally known as relative-clause 'donkey' sentences because of examples like

(5) every farmer who owns a donkey feeds it

Providing plausible truth conditions for such sentences formed one of the key successes for Discourse Representation Theory but, notoriously, it only achieved this by not assigning a binary structure to such sentences. The work in this thesis provides a dynamic semantics for binary structured quantified sentences. Most importantly, it does so by threading the output context for the first argument into the input context for the second argument, *i.e.* by simply using the mechanism which is distinctive of dynamic theories. Other dynamic theories have attempted to handle binary quantificational structures but they have all made essential use of some *other* mechanism besides the threading of contexts ([Kamp & Reyle *forth.*], [Chierchia 88],[van Eijck & de Vries 91]).

1.1 The Dynamic Conception of Meaning

Dynamic Logic arises naturally in Theoretical Computer Science ([Harel 84]). In order to study computer programs formally, one can consider programs, and parts of pro-

grams, to denote pairs of execution states of a computer. The execution of a program on a computer in a certain state results in a transition of the computer into another, possibly identical, state. The next program may then be executed in the state resulting from the previous execution and this will produce yet a third state. The process can be repeated an arbitrary number of times. For example, suppose we take an execution state to be the current assignment of values to the variables used by a program. The state in which '3' is assigned to variable i, 'a' to j and '1011' to k will be written $\{i \rightarrow 3, j \rightarrow a, k \rightarrow 1011\}$. Now consider the program which consists solely of the statement '`x := 1 + y`' (add the value of 1 to the value of variable '`y`' and assign the result to the variable '`x`'). This particular program denotes the set of pairs of execution states $< \{x \rightarrow a, y \rightarrow b\}, \{x \rightarrow c, y \rightarrow b\} >$, where $c = b + 1$. Once we have formalized a given program, we may be able to prove some useful properties of that program: for example, that it terminates and that the final execution state is always a certain interesting function of the initial execution state.

It is perhaps unsurprising that such a general conception should be thought capable of providing insights into the analysis of the interpretation of natural language sentences, as well as those of computer programs. Indeed, the idea finds a natural home in much work in computational semantics. In the realm of discourse interpretation, it has long been evident that determining the appropriate interpretation of certain expressions depends on many factors including not only context-invariant features of meaning, general knowledge about the world and its current state but also the previous discourse and how that discourse has been interpreted (*cf.* [Isard 75],[Grosz *et al* 86]).

Even in the rarefied atmosphere of formal semantics, examples of the general conception can be found well before the analogy between the dynamic interpretation of computer programs and that of natural language was stressed and exploited in 'Dynamic Predicate Logic' ([Groenendijk & Stokhof 91b], the paper has circulated since the mid-1980s in various guises). For example, the dependency of interpretation on contextual factors in general was highlighted early by the development of indexical

semantics. In indexical semantics, elements of the context capable of influencing interpretation are represented by indices with respect to which formulae are evaluated. For example, indices for world, time, place, agent and addressee were postulated. It does not seem against the spirit of this enterprise to have indices available for the objects already introduced by the current discourse - for example, a sequence of indices for the sequence of objects mentioned. Although this picture allows the possibility of dynamic interpretation, it is, as yet, a little short on the precise specification of how the contents of the contexts are to be determined by what has been said and, indeed, how the appropriate features of the context are to be selected for use in interpretation. As an example of an early use of dynamic interpretation, we can briefly consider Cooper's formal semantical account of 'pronouns of laziness' [Cooper 79]. Cooper considers the interpretation of a sentence which forms a classic case for dynamic semantics

(6) Every man who owns a donkey feeds it

and he makes essential appeal to the idea that sentences are evaluated with respect to a *context of use*, an idea promulgated though not fully exploited by [Montague 74b]. Cooper treats the pronoun 'it' as denoting the set of properties of a unique individual who stands in some contextually given relation to an individual. Call the relation 'R'. Cooper's analysis of 6 can be paraphrased as 'It is true of each man who owns a donkey that he feeds a unique individual that stands in relation R to him'. Cooper's suggestion is that the context of use can tell us that the relation in question is actually that of being a donkey owned by the denotation of 'x' (where 'x' is the variable over which the subject noun phrase quantifies and thereby denotes a man that owns a donkey). The resulting truth conditions for 6 are that every man who owns a donkey feeds the unique donkey that he owns. According to Cooper, it is the job of *pragmatics* to tell us what the context of use is but, presumably, whatever story is told here will invoke the fact that the phrase 'man who owns a donkey' has just been used and interpreted. It is otherwise quite mysterious how one could possibly have arrived at the right selection for what the relation R is.

Hans Kamp's Discourse Representation Theory (DRT) can not unfairly claim to be the first dynamic theory ([Kamp 81]), though similar ideas were developed almost contemporaneously by Heim in her thesis [Heim 82]. The theory has for its input fairly simple and standard surface syntactic structures upon the basis of which an algorithm builds a new structure called a Discourse Representation Structure (DRS). DRSs are provided with a model-theoretic semantics, in an orthodox fashion. The dynamic element arises from the algorithmic construction of the DRSs. Just as the question of which step a computer will take next is settled, in part, by examining the values of variables set by previous steps, so the question of what DRS to construct next is settled, in part, by what DRS has already been constructed. As a simple example, if the input sentence is 'he sat down' and no DRS construction has taken place yet, then the algorithm will terminate with no DRS having been built. This is intended to correspond to the fact that a discourse initial occurrence of 'he sat down' is uninterpretable. If a DRS has been built previously, as a result of processing 'A man entered the room' for example, then a further DRS may be built. This DRS will encode the fact that 'he' and 'a man' are anaphorically related and the model theoretic semantics will predict that the discourse is true just in case some man both entered the room and sat down.

It should be noted that, in DRT, the dynamic element is located entirely in the algorithmic transformations of DRSs. The model theoretic interpretation of DRSs is not itself dynamic. It makes no use of the notion of interpretation in a context or of interpretation inducing a change in the context. In this respect DRT lies close to work in computational semantics where input syntactic structures and discourse models are examined and manipulated in order to generate further representations which can themselves be interpreted rigorously (and sometimes not so rigorously) by the techniques of formal semantics.

Barwise's language $L(ss)$, developed in [Barwise 87], constitutes the first attempt, in the area of natural language interpretation, to locate the dynamic element in interpre-

tation firmly within the model-theoretic component. The two-stage procedure of DRT is replaced by a one-stage dynamic interpretation. The idea is taken up and developed in the system of 'Dynamic Predicate Logic' (DPL) [Groenendijk & Stokhof 91b] where a dynamic interpretation of the syntax of first order predicate logic is defined. The models for the dynamic interpretation are identical to those used for the standard static interpretation. In fact, the major difference between the two consists just in the fact that the key semantic notion in the standard interpretation 'satisfaction of a formula by an assignment of objects to variables' is replaced in the dynamic version by the notion 'satisfaction of a formula by a pair of assignments of objects to variables.' It is the use of pairs of assignments which entitles DPL to the title 'dynamic', because each pair can be thought of as an *input context* and an *output context* for a formula. The output context for one formula may become the input context for the next and information is thereby threaded between formulae, rather than simply being passed down hierarchically from a formula to its subformulae as one might expect from more standard semantic theories. Whilst it is threading that makes DPL dynamic, what justifies it to the title 'semantics' is that the theory also provides a characterization of the circumstances in which discourses are *true*. In the standard interpretation of first order predicate logic (this I will subsequently refer to by 'FOPL', meaning that language *as* standardly interpreted and not just its syntax), the satisfaction relation between formulae and assignments provides, as a special case, for a definition of the property of truth of closed formulae. Closed formulae are formulae with no unbound variables. Similarly, in DPL, the relation between pairs of assignments and formulae can also be used to define the truth *simpliciter* of a (closed) discourse as a special case. DPL is therefore justifiably called a dynamic semantics. The 'double-aspect' feature of dynamic logic (relating context change and truth conditional content) is a particularly attractive one and forms a central point of interest for [van Bentham 91]. Indeed, we might not unreasonably draw the contrast that whereas in DRT dynamism and truth conditional content are expressed in a double-stage theory, in DPL they are expressed

in a double-aspect theory.

1.2 Plan of the Thesis

Since the principle advantage claimed for DPL over other dynamic theories of discourse interpretation, notably DRT, is that DPL is *compositional*, chapter 2 gives formal analyses for both DRT and DPL before discussing them both with respect to this fundamental and highly controversial topic. During this discussion I shall nail my own colours to the mast - or at least specify which colours should be nailed how far up which mast. The results of the discussion are used to motivate features of the semantic definitions which appear in subsequent chapters. These features include the use of no intermediate level of representation, thereby avoiding the threat of a psychological interpretation for that level, and also the use of top-down truth definitions, these being perfectly compositional in the intuitive sense. The top-down feature also proves useful in giving a semantics to deal with quantifier scope ambiguities. The discussion also highlights the importance of defining a precise dynamic semantics for actual English, rather than for a formal language such as the syntax of first order logic.

In Chapters 3 and 4, we precisely define a dynamic semantics for English surface structures. In this way, we emulate DRT whose input also consists of syntactically structured English sentences and those structures also do not include the information about what antecedents pronouns have. In contrast, in the formalized language DPL, those decisions have already been taken. In chapter 3 a semantics for English structures which are ambiguous with respect to quantifier scope is defined. The approach takes its lead from the quantifier scoping algorithm of [Hobbs & Shieber 87]. Conditionals rather than the more standard bi-conditionals are used in the semantic analysis and an improved quantifier scoping algorithm is described. The approach is also compared to others in the literature including those of [Montague 74a], [Cooper 83], [Hendriks 90] and [Emms 90]. The account is compositional in the sense of chapter 2. In chapter

4, the approach is extended to cope with structures where pronouns have yet to be resolved. Theories such as DPL have not been forthcoming on how one may decide what a possible antecedent for a discourse pronoun is. Furthermore, it looks implausible to appeal to syntax in order to license co-indexing. Traditional syntactic theories have appealed to relations in syntactic structure but these relations do not hold across whole discourses and it is discourse anaphora that dynamic logic is primarily concerned with. A properly dynamic theory ought to stress the role of the input context in selecting possible antecedents, and the role of the output context in making certain antecedents available and others not so. A theory is proposed which makes use of both of these notions. Syntactic restrictions on anaphora are stated using a version of Linking Theory ([Higginbotham 83]) which also stresses the asymmetric nature of the link between an antecedent and its pronominal dependent. The asymmetric nature of this link had been stressed in Barwise's early version of dynamic logic ([Barwise 87]).

In chapter 5, an algorithm faithful to the interpretation procedure of chapters 3 and 4 is described and compared with some other recent work in computational semantics which has also proposed methods for handling the same phenomena and which also make use of the notion of a discourse context which is threaded through the evaluation of discourse ([Latecki & Pinkal 90], [Pereira & Pollack 91] and [Alshawi 90]).

Although DPL was designed to emulate DRT empirically, both theories have needed to take account of other developments in the field of formal semantics. Recent work in DRT, for example, has proposed a method for incorporating binary structured quantifiers (*cf.* [Barwise & Cooper 82]), *i.e.* quantifiers that take two arguments rather than just one [Kamp & Reyle *forth.*]. Chapter 6 develops a new extension of DPL to include binary structured quantifiers. In particular, in chapter 6 we are concerned to construct a theory in which information from the first argument is genuinely *threaded* into the second argument, since it this notion which supplies the central idea of dynamism in DPL. Other accounts of generalized quantifiers in the dynamic environment (*e.g.* [Chierchia 88], [Kamp & Reyle *forth.*],[van Eijck & de Vries 91]) have been given

previously but these all make use of *other* operations besides threading in order to secure links between the two parts of a binary structure. As a result, the original idea from DRT that inter-sentential and extra-sentential anaphora are handled by the same mechanism is lost. The theory of chapter 6 restores the link.

Chapter 7 draws some conclusions from the work and discusses some possible future directions. The role of *incremental* interpretation in dynamic logic is also discussed. Incremental interpretation seems to be a natural ally of dynamic interpretation but it does not follow simply from adopting from the dynamic approach and incorporating both dynamism and incrementality may turn out to be a significantly more complex problem.

Chapter 2

Compositionality in DPL and DRT

Dynamic Predicate Logic is 'a first step towards a compositional, non-representational theory of discourse semantics' ([Groenendijk & Stokhof 91b]). The theory is designed to cover the same empirical data as the earliest version of Discourse Representation Theory ([Kamp 81]) but to keep to the principle of compositionality which is upheld on methodological, computational and philosophical grounds. In particular, Groenendijk and Stokhof are concerned to have a theory which does not postulate a psychologically real level of semantic representation apart from 'syntactic structure and meaning proper'.

The first two sections of this chapter give a formal, though brief, presentation of DRT and then a presentation of DPL together with some worked examples to show precisely how the dynamic effects are produced in the system. With this background there then follows a discussion of the principle of compositionality itself.

2.1 Discourse Representation Theory

The following definitions provide a formal account of a simple version of Discourse Representation Theory based on, though not identical to, that of [Kamp & Reyle *forth.*]. The definition contains three main parts. The first is a simple sentence grammar which

provides the structuring for the input strings comprising a discourse. The second is the DRS construction algorithm which constructs DRSs from a sequence of structured sentences. The third is the model theoretic semantics for the resulting DRSs.

2.1.1 Simple sentence grammar

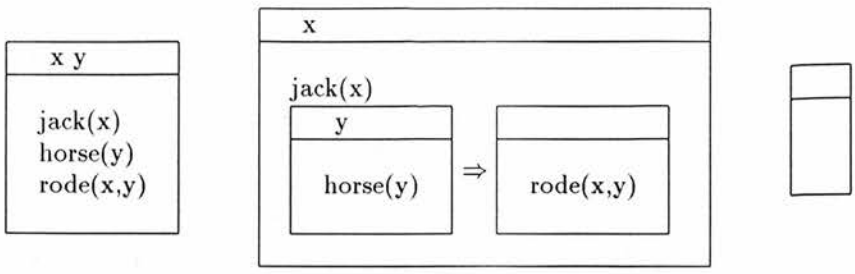
A grammar consists of a lexicon and a set of syntactic rules. The lexicon V is the union of the following sets $V_{pn}, V_n, V_{tv}, V_{iv}, V_{det}, V_{pro}$, whose membership I shall leave unspecified.

Lexical Insertion	$X \rightarrow \alpha$ where $X \in \{pn, n, tv, iv, det, pro\}$ and $\alpha \in V_X$			
Rules	S	$\rightarrow$ NP VP,	VP	$\rightarrow$ TV NP
	NP	$\rightarrow$ DET N,	NP	$\rightarrow$ PN
	NP	$\rightarrow$ PRO,	VP	$\rightarrow$ IV

2.1.2 Definition of a DRS

A DRS K confined to a lexicon V and set of discourse referents R consists of a set of discourse referents U_K ($U_K \subseteq R$) and a set of *DRS conditions* Con_K each confined to V and R. A *DRS condition* (confined to V and R) is either an atomic DRS condition or a complex DRS condition. An atomic condition is one of the following expressions: $pn(x)$, $n(x)$, $iv(x)$, $tv(x, y)$ where $x, y \in R$, $pn \in V_{pn}$, $n \in V_n$, $iv \in P_{iv}$, $tv \in P_{tv}$. A complex DRS condition is $K_1 \Rightarrow K_2$, where K_1 and K_2 are both DRSs confined to V and R.

Using the traditional box notation for DRT where the set of discourse referents are put in a box directly above the box containing the DRS conditions, we find the following are all well-defined DRSs (assuming suitable membership of V and R), with the last being the *empty DRS*.



2.1.3 From Discourses to DRSs

In DRT, a discourse is simply a sequence of (structured) sentences $S_0, S_1, \ldots, S_n$. The DRS construction algorithm is then specified as follows

DRS construction algorithm

initialize proto-DRS d to the *empty DRS*
repeat from $c = 0$ until $c = n$
  add S_c to Con_d
  repeat until d is irreducible
    update d by applying a DRS construction rule
  end-repeat
end-repeat

The 'proto-DRSs' referred to in the algorithm will not, in general, be DRSs according to the definition of 2.1.2, although they resemble them. The reason is that inside a proto-DRS one may find structures other than those defined in 2.1.2. For example, a proto-DRS will contain syntactic structures generated by the grammar (by the step 'add S_c to Con_d') but these are not valid DRS conditions. It is the objective of the algorithm to process proto-DRSs until a DRS is obtained - an irreducible proto-DRS is therefore intended to be a DRS.

The DRS construction rules each consist of a *triggering configuration* and a set of *actions*. A triggering configuration is a syntactic structure, which may contain variables, and can thereby pattern-match with other syntactic structures. Given a proto-DRS d containing a syntactic structure s, if the triggering configuration pattern-matches with s, then the rule is *applicable* and the *actions* may be executed. The actions will generally destructively modify the proto-DRS d. DRS construction rules are selected

and applied non-deterministically. A proto-DRS for which no DRS construction rule is applicable is said to be *irreducible*. A simple example set of rules is given below. The rules are just those of [Kamp & Reyle *forth.*] except as stated.

For the purposes of pronoun resolution, DRT makes use of a relation between discourse referents and conditions in proto-DRSs called 'accessibility'. This is defined in terms of 'subordination' ($\leq$) between proto-DRSs.

Immediate Subordination ($\prec$)

$K_1 \prec K_2$ if either

i) $K_1 = K_2$

ii) $\exists K_3$ ('$K_2 \Rightarrow K_1$' $\in Con_{K_3}$)

iii) $\exists K_3$ ('$K_1 \Rightarrow K_3$' $\in Con_{K_2}$)

Subordination ($\leq$)

$K_1 \leq K_2$ if either

i) $K_1 \prec K_2$

ii) $\exists K_3 \, (K_1 \prec K_3 \wedge K_3 \leq K_2)$

Let C be a condition of a proto-DRS K. A discourse referent x is accessible from C in K if there is a proto-DRS K_1 such that i) $x \in U_{K_1}$ and ii) $K \leq K_1$.

For each of the rules stated below, d refers to the input proto-DRS and s to the structure with which the trigger is pattern-matched.

DRS construction rules

CR.PN triggers: $[_S\ [_{NP}\ [_{PN}\ \alpha]]\ [_{VP}]]$ or
$[_{VP}\ [_V]\ [_{NP}\ [_{PN}\ \alpha]]]$
actions: add new discourse referent x into U_d
add $\alpha(x)$ into Con_d
substitute x for $[_{NP}\ [_{PN}\ \alpha]]$ in s

CR.PR triggers: $[_S\ [_{NP}\ [_{PRO}\ \alpha]]\ [_{VP}]]$ or
$[_{VP}\ [_V]\ [_{NP}\ [_{PRO}\ \alpha]]]$
actions: choose an accessible discourse referent x
substitute x for $[_{NP}\ [_{PRO}\ \alpha]]$ in s

CR.ID triggers: $[_S\ [_{NP}\ [_{DET}\ a]\ [_N\ \alpha]]\ [_{VP}]]$ or
$[_{VP}\ [_V]\ [_{NP}\ [_{DET}\ a]\ [_N\ \alpha]]]$
actions: add new discourse referent x into U_d
add new condition $\alpha(x)$ ito Con_d
substitute x for $[_{NP}\ [_{DET}\ a]\ [_N\ \alpha]]$ in s

CR.EV triggers: $[_S\ [_{NP}\ [_{DET}\ every]\ [_N\ \alpha]]\ [_{VP}]]$ or
$[_{VP}\ [_V]\ [_{NP}\ [_{DET}\ every]\ [_N\ \alpha]]]$
actions: add new condition $K_1 \Rightarrow K_2$ (K_1, K_2 are both empty DRSs)
add new discourse referent x into U_{K_1}
add new condition $\alpha(x)$ into Con_{K_1}
generate γ by substituting x for $[_{NP}\ [_{DET}\ every]\ [_N\ \alpha]]$ in s
add γ into Con_{K_2}
delete s from Con_d

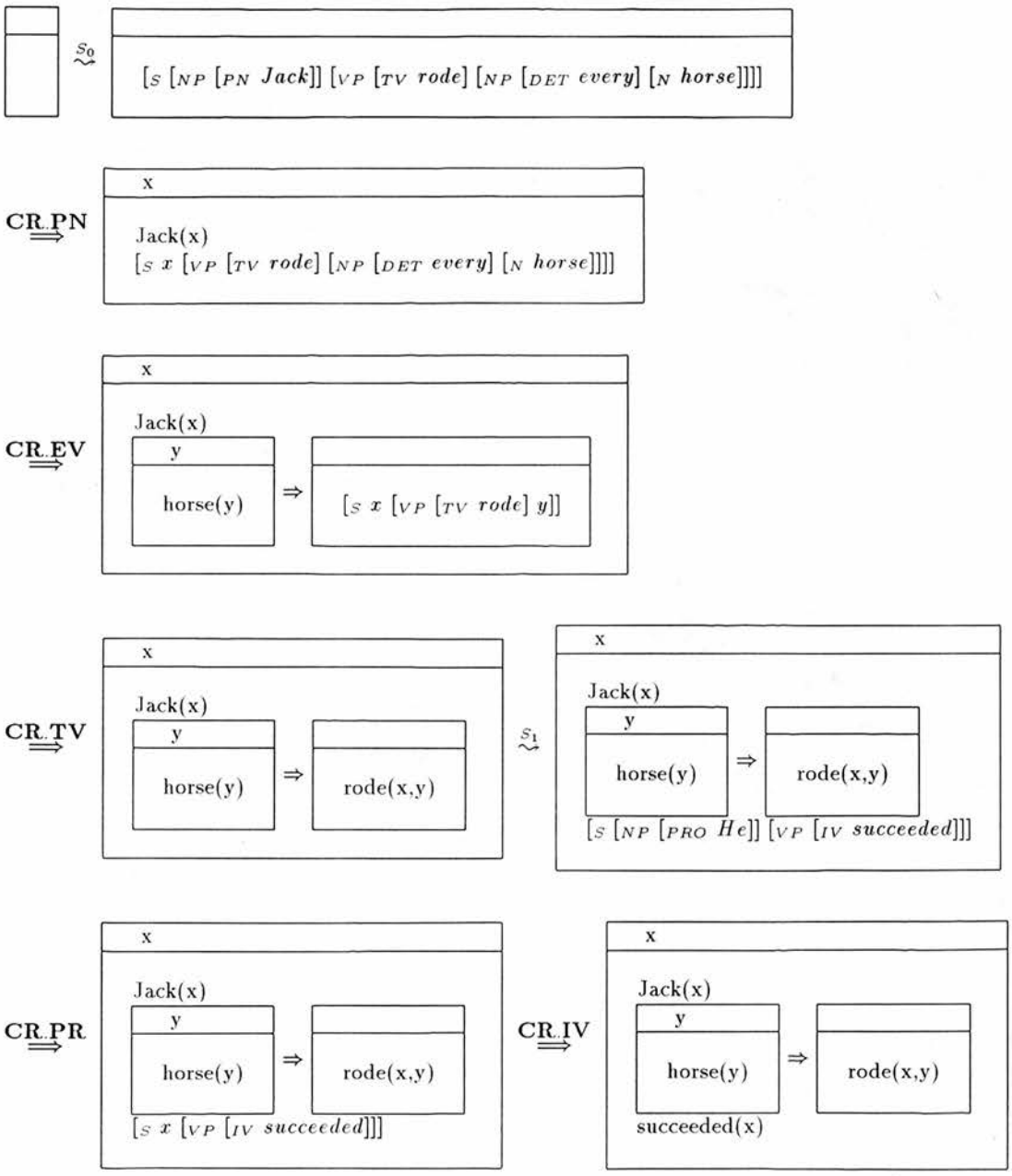
CR.IV triggers: $[_S\ x\ [_{VP}\ [_{IV}\ \alpha]]]$ $(x \in U_d)$
actions: add new condition $\alpha(x)$ into Con_d
delete s from Con_d

CR.TV triggers: $[_S\ x\ [_{VP}\ [_{TV}\ \alpha]\ y]]$ $(x, y \in U_d)$
actions: add new condition $\alpha(x, y)$ into Con_d
delete s from Con_d

The only differences with the rules of [Kamp & Reyle *forth.*] are in rule **CR.PR** where Kamp and Reyle introduce a new discourse referent which is then stated to be identical with the chosen accessible discourse referent. Also, Kamp and Reyle treat a DRS condition such as 'x saw y' simply as an abbreviation for $[_S\ x\ [_{VP}\ [_{TV}$ saw$]\ y]]$ whereas I have inserted a step transforming the latter explicitly syntactic structure into a DRS condition. These minor differences do not affect the issues to be discussed here.

Example

Consider the discourse S_0, S_1 where S_0 is 'Jack rode every horse' and S_1 is 'He succeeded'. The diagram below illustrates how the DRS construction algorithm processes the discourse. The symbol $\overset{S_n}{\rightsquigarrow}$ corresponds to adding the nth sentence into the proto-DRS and $\overset{\text{CR.RULE}}{\Longrightarrow}$ corresponds to the execution of the particular DRS construction rule named **CR.RULE.**



2.1.4 Interpreting DRSs in a model

The objective of the DRS construction algorithm is to construct a (non-proto) DRS. DRSs are interpreted model-theoretically and the discourse from which the DRS is derived is also thereby interpreted.

A DRT-model M for a lexicon V is a pair $< A, I >$ where A is a set of individuals and I is a function from members of P_{pn} to members of A, from members of P_n, P_{iv} to subsets of A and members of P_{tv} to sets of pairs of members of A. An *embedding* is a partial function from R (a set of discourse referents) to members of A. One embedding g extends another f by a DRS K ($g \supseteq_K f$) just in case $dom(g) = dom(f) \cup U_K$ and g and f agree on the values of all discourse referents in $dom(g) \cap dom(f)$. The key concepts to be characterized recursively are those of an embedding f verifying a DRS condition and a DRS, with respect to a model M. I shall omit reference to M and write 'f verifies K' as $[\![K]\!]^f$.

DRS Verification

i) $[\![K]\!]^f$ (K is a DRS) iff $\forall_{x \in Con_K} [\![x]\!]^f$

DRS Condition Verification

ii) $[\![\alpha(x)]\!]^f$ ($\alpha \in P_{pn}$) iff $f(x) = I(\alpha)$

iii) $[\![\alpha(x)]\!]^f$ ($\alpha \in P_{iv} \cup P_n$) iff $f(x) \in I(\alpha)$

iv) $[\![\alpha(x, y)]\!]^f$ ($\alpha \in P_{tv}$) iff $< f(x), f(y) > \in I(\alpha)$

v) $[\![K_1 \Rightarrow K_2]\!]^f$ iff $\forall g \, ((g \supseteq_{K_1} f \wedge [\![K_1]\!]^g) \rightarrow \exists h \, (h \supseteq_{K_2} g \wedge [\![K_2]\!]^h))$

In order to define the notion of 'truth in a model' we first define the *proper* or *closed* DRSs. A discourse referent x is *free* in a DRS K just in case $x \notin U_K$ and x is free in a DRS condition C ($C \in Con_K$). A discourse referent x is free in an atomic DRS condition if it occurs in that condition and x is free in '$K_1 \Rightarrow K_2$' just if either x is free in K_1 or it is both free in K_2 and there is no DRS K_3 such that $K_2 \leq K_3$ and $x \in U_{K_3}$.

A DRS K is *proper* just in case no discourse referent is *free* in K.

Truth in a model

A proper DRS K confined to V and R is true in a model $M =< A, I >$ for the lexicon

V just in case *there is* an embedding f from U_K into A such that $[\![K]\!]^f$.

Example revisited

Our example discourse was 'Jack rode every horse. He succeeded'. Consider a model M_1 in which the set A contains Jack and a number of horses. Suppose too that I assigns Jack to 'Jack', the set of horses to 'horse' and the set of succeeding objects to 'succeeded'. The DRS we derived for the discourse is true in M_1 just in case there is an embedding f from {x} into A such that all the conditions are verified by f. The first condition is 'Jack(x)', which is verified by f iff 'x' is assigned to Jack. The second condition is verified by f iff every embedding from {x,y} which assigns Jack to 'x' and verifies 'horse(y)' also verifies 'rode(x,y)'. That is, it is true of every horse that Jack rode it. The third condition is 'succeeded(x)', which is verified by f just in case Jack is one of the succeeding objects. Overall, the discourse is true in M_1 just in case Jack rode every horse and succeeded. Those are the correct truth conditions for the discourse.

2.1.5 Summary

The following salient features of DRT can be noted from our exposition. First, DRT provides truth conditions for discourses (including extra-sentential anaphora) and it does so via a two-stage procedure. The first stage consists of building a DRS by the DRS construction algorithm and the second consists of interpreting the DRS in a model by the rules for DRS verification. The algorithm builds the DRS by recursively breaking down a syntactic structure. All occurrences of pronouns are handled by just one rule **CR.PR**. The value of a pronoun is determined by the accessibility relation between discourse markers and DRS conditions and not, for example, by a relation between the pronoun and another noun phrase in the input syntactic structure. Finally, indefinite noun phrases are treated rather like referring expressions by introducing discourse referents but the quantificational force associated with an indefinite depends on the

structure of the discourse.

2.2 Dynamic Predicate Logic

In this section, a formal exposition of a simple version of DPL is given. The exposition is similar to that of [Groenendijk & Stokhof 91b] except that it uses partial assignments instead of total assignments. One advantage of this is that it thereby resembles the truth definition for DRSs which also uses partial functions. In particular, we will be able to answer one of Kamp's questions about DPL namely 'what assignment could correspond to the minimal information state, that in which no information is available ?' ([Kamp 90]). The answer to that will be simply the empty assignment. A formula will be true *simpliciter* if it is true with respect to the empty assignment.

Since DPL is a *one-stage* theory, the theory consists of defining first the syntax of the language of DPL and then its semantics. The definitions will be followed by two worked examples to illustrate the distinctive features of the dynamic semantics.

2.2.1 DPL Syntax

The syntax of DPL is just that of that most familiar language: FOPL. Formally, DPL consists of a set of variables and n-ary predicates, together with the logical constants: negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$) plus existential and universal quantification ($\exists$ and $\forall$). The set of formulae of DPL is inductively defined by the following rules: if $t_1, \ldots, t_n$ are variables and P is an n-ary predicate, then $P(t_1, \ldots, t_n)$ is a formula; if ϕ is a formula, then so is $\neg(\phi)$; if ϕ and ψ are formulae then so are $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$; if ϕ is a formula and x is a variable, then $\exists x(\phi)$ and $\forall x(\phi)$ are formulae too.

Free variables

I shall use partial assignments in the semantic definitions and therefore need to

guarantee that the meta-language expression 'the value of x according to assignment g' is not used when g is undefined for x. This will be achieved by stipulating that the clauses defining $[\![\phi]\!]^{i\ o}$ – that ϕ is true with respect to i and o – apply just in case i is defined on all the *free variables* of ϕ.

$FV(\phi)$ denotes the set of *free variables* in ϕ and $AQ(\phi)$ denotes the set of variables actively quantified over in ϕ. The notion of 'actively quantified over' corresponds to those variables existentially quantified over in a formula λ and which, should they occur elsewhere in a formula containing λ, may be semantically dependent on the quantifier in λ (*cf.* [Groenendijk & Stokhof 91b]) . According to the definitions below, no existential quantifier in a sub-formula of form $\neg(\psi)$, $\psi \vee \pi$, $\psi \rightarrow \pi$ and $\forall \mu(\psi)$ can bind a variable occurring in a formula containing that sub-formula. Having said this, it is perhaps easiest to regard $AQ(\phi)$ simply as a subsidiary technical notion used in giving a definition of $FV(\phi)$.

1. If ϕ is of form $P(t_1, \ldots, t_n)$,
   $FV(\phi)$ is the set of all variables occurring in ϕ
   $AQ(\phi) = \emptyset$.
2. If ϕ is of form $\neg(\psi)$,
   $FV(\phi) = FV(\psi)$
   $AQ(\phi) = \emptyset$
3. If ϕ is of form $(\psi \wedge \pi)$,
   $FV(\phi) = FV(\psi) \cup \{\mu \mid \mu \in FV(\pi), \mu \notin AQ(\psi)\}$
   $AQ(\phi) = AQ(\psi) \cup AQ(\pi)$
4. If ϕ is of form $(\psi \vee \pi)$,
   $FV(\phi) = FV(\psi) \cup FV(\pi)$
   $AQ(\phi) = \emptyset$
5. If ϕ is of form $(\psi \rightarrow \pi)$,
   $FV(\phi) = FV(\psi) \cup \{\mu \mid \mu \in FV(\pi), \mu \notin AQ(\psi)\}$
   $AQ(\phi) = \emptyset$
6. If ϕ is of form $\exists\mu(\psi)$,
   $FV(\phi) = FV(\psi) - \{\mu\}$
   $AQ(\phi) = AQ(\psi) \cup \{\mu\}$
7. If ϕ is of form $\forall\mu(\psi)$,
   $FV(\phi) = FV(\psi) - \{\mu\}$
   $AQ(\phi) = \emptyset$

A *DPL-discourse* is defined to be a formula with no free variables.

2.2.2 DPL Semantics

A model for DPL is a pair $< D, F >$ where D is a non-empty set of individuals and F is a function from n-ary predicates to n-ary relations on D. An assignment is a partial function from variables to members of D. Let $\emptyset$ be the null assignment, that assignment with an entirely empty domain. If g is an assignment, then g_x^α is g extended or modified so that the object α is assigned to the variable x. For example, $\emptyset_z^\beta$ is the partial function which assigns β to z and is undefined everywhere else. The value of a variable with respect to an assignment $[\![x]\!]^g$ is $g(x)$ for all variables x. $[\![\phi]\!]^{i\ o}$ means 'ϕ is true with respect to i and o' and is defined just in case i is defined on all the free variables of ϕ. The meta-language used to state the following rules is FOPL.

1. $[\![P(t_1, \ldots, t_n)]\!]^{i\ o}$ iff $i = o \wedge < [\![t_1]\!]^i, \ldots, [\![t_n]\!]^i > \in F(P)$
2. $[\![\neg(\phi)]\!]^{i\ o}$ iff $i = o \wedge \neg\ \exists k\ ([\![\phi]\!]^{i\ k})$
3. $[\![(\phi \wedge \psi)]\!]^{i\ o}$ iff $\exists k\ ([\![\phi]\!]^{i\ k} \wedge [\![\psi]\!]^{k\ o})$
4. $[\![(\phi \vee \psi)]\!]^{i\ o}$ iff $i = o \wedge \exists k\ ([\![\phi]\!]^{i\ k} \vee [\![\psi]\!]^{i\ k})$
5. $[\![(\phi \rightarrow \psi)]\!]^{i\ o}$ iff $i = o \wedge \forall k\ ([\![\phi]\!]^{i\ k} \rightarrow \exists l\ ([\![\psi]\!]^{k\ l}))$
6. $[\![\exists x\ \phi]\!]^{i\ o}$ iff $\exists \alpha\ ([\![\phi]\!]^{i_x^\alpha\ o})$
7. $[\![\forall x\ \phi]\!]^{i\ o}$ iff $i = o \wedge \forall \alpha\ (\exists k\ ([\![\phi]\!]^{i_x^\alpha\ k}))$

A DPL-discourse is defined to be *true* with respect to an assignment i just in case it is true with respect to $< i, k >$, for some k and it is true *simpliciter* if it is true with respect to $\emptyset$. That is 'ϕ is true iff $\exists k\ ([\![\phi]\!]^{\emptyset\ k})$'.

2.2.3 Examples

To give the flavour of the system, I shall work through two examples. In working through them, it is important to bear in mind the two key aspects of dynamic logic. First, formulae are interpreted with respect to two assignment functions, instead of just one as in FOPL and DRT. This feature allows the threading of information between the

evaluation of formulae. Secondly, the evaluation of a formula representing a discourse is aimed always at generating a statement of the truth conditions for that discourse.

First Example - $\exists x\ (m(x))\ \wedge\ w(x)$

According to the FOPL, the interpretation of the occurrence of x in $w(x)$ is quite unrelated to that of the occurrence of x in $m(x)$. In DPL however, there is a relation between the two occurrences and this will be demonstrated by a derivation of the truth conditions.

The following derivation proceeds top-down and step by step through the formula applying one semantic rule at each stage. The number of the rule used is given in square brackets like this: [1].

$$[\![\ \exists x\ (m(x))\ \wedge\ w(x)\ ]\!]^{i\ o} \text{ iff } \ldots$$

a) [3] $\exists k\ ([\![\ \exists x\ (m(x))\ ]\!]^{i\ k}\ \wedge\ [\![\ w(x)\ ]\!]^{k\ o}\ )$

b) [6] $\exists k\ (\exists \alpha\ ([\![\ m(x)\ ]\!]^{i^{\alpha}_{x}\ k})\ \wedge\ [\![\ w(x)\ ]\!]^{k\ o}\ )$

c) [1] $\exists k\ (\exists \alpha\ (i^{\alpha}_{x}\ =\ k\ \wedge\ [\![\ x\ ]\!]^{i^{\alpha}_{x}}\ \in F(m))\ \wedge\ [\![\ w(x)\ ]\!]^{k\ o}\ )$

d) [1] $\exists k\ (\exists \alpha\ (i^{\alpha}_{x}\ =\ k\ \wedge\ [\![\ x\ ]\!]^{i^{\alpha}_{x}}\ \in F(m))\ \wedge\ k\ =\ o\ \wedge\ [\![\ x\ ]\!]^{k}\ \in F(w))$

Noting that $[\![\ x\ ]\!]^{i^{\alpha}_{x}}$ is just α and that since $k\ =\ o$ we can drop the initial quantifier over k, the formula $k\ =\ o$ and replace all occurrences of k by occurrences of o, we obtain

$$[\![\ \exists x\ (m(x))\ \wedge\ w(x)\ ]\!]^{i\ o} \text{ iff } \ldots$$

e) $\exists \alpha\ (i^{\alpha}_{x}\ =\ o\ \wedge\ \alpha\ \in F(m))\ \wedge\ [\![\ x\ ]\!]^{o}\ \in F(w)$

At this point it is tempting to think that we can replace $[\![\ x\ ]\!]^{o}$ simply by α since $o\ =\ i^{\alpha}_{x}$ and $[\![\ x\ ]\!]^{i^{\alpha}_{x}}\ =\ \alpha$. However, we cannot do this because the *meta-language* is ordinary non-dynamic FOPL and the substitution would leave the new occurrence of α outside the scope of the quantifier that is supposed to be binding it. Such a substitution would result in the following statement

$[\![\exists x\ (m(x))\ \wedge\ w(x)]\!]^{i\ o}$ iff ...

e′) $\exists\alpha\ (i_x^{\alpha}\ =\ o\ \wedge\ \alpha\ \in F(m))\ \wedge\ \alpha\ \in F(w)$

but e′ is not equivalent to e for the reason just stated. This phenomenon shows precisely the difference in power between the object and meta-languages. The (non-dynamic) meta-language quantifiers have restricted scope compared to the (dynamic) object language quantifiers. (We could avoid this by rewriting the clauses in a dynamic meta-language - once we have understood how such dynamic languages work - but our present objective is to understand them). It is the threading of information which gives the dynamic language its additional binding properties. The Output context for $\exists x\ (m(x))$ becomes the Input context for $w(x)$. Any Output context k for $\exists x\ (m(x))$ determines as the value of x, *one particular* object that satisfies the predicate m. The important point is that there is a quantifier governing k which has wider scope than the quantifier actually introduced by the object language existential quantifier ($\exists\alpha$ by step b). Within the scope of that introduced quantifier we have a statement of form '$o\ =\ i_x^{\alpha}$'. This enables us to refer to α via the expression $o(x)$ even when we are outside of the scope governing α itself. It is rather like saying 'there is a man whom we may call 'John" and then going on to use 'John' elsewhere. In our example, the meta-language expression $o(x)$ plays the role of 'John'. We know that $o(x)$ denotes an object which is a man, just as 'John' does and we can also use $o(x)$ outside the formula '$\exists\alpha(o\ =\ i_x^{\alpha})$' just as we can use 'John' in subsequent sentences.

Now that we have seen how the dynamic existential quantifier can yet bind variables beyond the scope of an ordinary one, we can note that e is, in fact, equivalent to f below (in DPL and FOPL) which contains just one quantifier with scope over the whole formula.

f) $\exists\alpha\ (\alpha\ \in F(m)\ \wedge\ \alpha\ \in F(w))$

Nevertheless, it would be an error to think of dynamic logic as providing a 'wide-scope' device for the existential quantifier. There is no sense in which the existential quantifier

is 'moving' or being 'quantified in' at the level of the whole formula. The object language quantifier $\exists$ introduces into the meta-language (which is standardly interpreted) an ordinary non-dynamic existential quantifier. It is the device of 'threading' and interaction with the other logical constants (in this case, conjunction) which generates the dynamic effect. In this particular example, the result happens to be equivalent to a formula containing a single wide-scope existential quantifier. In general, this need not happen and the second worked example will illustrate a case where it does not happen.

Second Example - $\exists x\ (m(x)) \rightarrow w(x)$

As in the last example, the occurrence of x in $w(x)$ is, according to FOPL, unrelated to the earlier existential quantifier. In DPL, not only is there a relation but, if one were to express the result in FOPL, then one would most naturally do so by using a (wide-scope) universal quantifier and not an existential quantifier at all.

$[\![\exists x\ (m(x)) \rightarrow w(x)]\!]^{i\ o}$ iff ...

a) [5] $i = o \wedge \forall k\ ([\![\exists x\ (m(x))]\!]^{i\ k} \rightarrow \exists l([\![w(x)]\!]^{k\ l}))$

b) [6] $i = o \wedge \forall k\ (\exists \alpha\ ([\![m(x)]\!]^{i^{\alpha}_{x}\ k}) \rightarrow \exists l([\![w(x)]\!]^{k\ l}))$

c) [1] $i = o \wedge \forall k\ (\exists \alpha\ (i^{\alpha}_{x} = k \wedge [\![x]\!]^{i^{\alpha}_{x}} \in F(m)) \rightarrow \exists l([\![w(x)]\!]^{k\ l}))$

d) [1] $i = o \wedge \forall k\ (\exists \alpha\ (i^{\alpha}_{x} = k \wedge [\![x]\!]^{i^{\alpha}_{x}} \in F(m)) \rightarrow \exists l(k = l \wedge [\![x]\!]^{l} \in F(w)))$

Once more, this can be simplified. $[\![x]\!]^{i^{\alpha}_{x}}$ is just α. We can also drop the quantification over l and replace all occurrences of l by k since k and l are identical.

$[\![\exists x\ (m(x)) \rightarrow w(x)]\!]^{i\ o}$ iff ...

e) $i = o \wedge \forall k\ (\exists \alpha\ (i^{\alpha}_{x} = k \wedge \alpha \in F(m)) \rightarrow [\![x]\!]^{k} \in F(w))$

So, our formula is true just in case every assignment k which assigns an object in the extension of m to x assigns an object in the extension of w to x. That is, is every object in the extension of m in the extension of w ? Once more, we may note that we cannot substitute α for $[\![x]\!]^{k}$ even though k is i^{α}_{x} since α would occur outside the scope of its intended binder. Essential use has again been made of identity within

a (narrowly scoped) quantified formula together with the threading of information between formulae. In this case, the information is universally quantified over and this is as a result of interaction with the non-standard interpretation of another logical constant, implication. We could therefore rewrite e as f

$$[\![(\exists x\ (m(x)) \rightarrow w(x))]\!]^{i\ o} \text{ iff } \ldots$$

f) $i = o \wedge \forall \alpha\ (\alpha \in F(m) \rightarrow \alpha \in F(w))$

but this rewrite does not display the essential dynamic element used in its generation.

It is worth examining briefly the 'indefinites as variables' thesis often held to be characteristic of DRT (*e.g.* [Heim 90]). The thesis is that indefinite noun phrases simply introduce discourse referents and do not themselves have quantificational import. Any quantificational force arises from quantifiers associated with the structure of the DRS in which the discourse referents occur. At first sight, DPL is quite different because indefinite noun phrases *do* appear in DPL as existential quantifiers after all. However, the impression is rather misleading. In our first example, $\exists x\ (m(x)) \wedge w(x)$ there is an existential quantifier present and its semantics is given in the meta-language by using existential quantification. Nevertheless, the effect of this quantification is subsumed by another quantifier with wider scope which is quantifying over exactly the same thing. This is achieved by the special use of identity within the scope of a quantifier. Just as $\exists x\ (x = \text{Socrates} \wedge m(x))$ actually has no existential force resulting from the use of $\exists$, so the same is true of our two examples. In both, the existential quantifiers present in the object language do not end up contributing existential force to the semantics of the formula. In each case they existentially quantify over something which is identical to some other thing quantified over by a higher quantifier. So while indefinites may be translated as existential quantifiers and are not 'simply' variables, the presence of the existential quantifier is not particularly significant.

2.2.4 Summary

DPL provides alternative truth conditions for the formulae of FOPL. DPL uses a simple one-stage procedure (a truth definition) which makes essential use of the notion 'satisfaction of a formula by a pair of assignments'. Each pair of assignments can be thought of as an input context and an output context. The key dynamic effects of the system are generated by the use of threading, *i.e.* by making the output context for one formula become the input context for the next. The value of any variable, as in FOPL, depends on what object is assigned to it by an assignment function. All occurrences of variables are handled by the same rule.

2.3 Compositionality

Compositionality, or 'Frege's Principle', is the thesis that 'the meaning of the whole is a function of the meaning of the parts and the way they are put together'. Although the principle is often attributed to Frege, similar ideas had, as is well known, been discussed and argued about much earlier, for example, by early Indian grammarians (for a recent article see [Matilal & Sen 90]). At a general level, as Partee remarks ([Partee 84]), the thesis looks quite uncontroversial. If you know the meanings of 'John', 'saw' and 'Mary' and you know the significance of their construction in 'John saw Mary', then you are in a position to know the meaning of 'John saw Mary' itself. Nevertheless, the proper fleshing out of the principle in formal detail is contentious.

One particularly strong realisation of the principle is found in Montague's programme of Universal Grammar ([Montague 74b]) - strong in the sense that several conditions are imposed. In Montague's system, the syntax (of a disambiguated language) is characterized by an algebra, that is a set of objects and a set of operations on those objects. The semantics is also given by an algebra (semantic objects plus semantic operations) and the interpretation of the language is given by a *homomorphism* from the former into the latter. The resulting system has the following important properties.

First, given any syntactic part (according to the syntactic algebra), we can find some one object that is its meaning. Since a homomorphism is a total function, we are guaranteed both that there is a semantic object which is the meaning of the syntactic object and that there is only one. Secondly, given any syntactic operation we can also find out what significance that operation has - for the homomorphism guarantees a corresponding semantic operation. Finally, once we have the meanings of our syntactic parts and the significance of their construction, then the meaning of the whole is indeed immediately and uniquely determined. It is often remarked that even this strong system of compositionality does not by itself have empirical consequences - for there are still many degrees of freedom left open, for instance in what the contents of the syntactic and semantic algebras may be (*cf.* [Janssen 83]).

Although no-one doubts that Montague's system is compositional (in the intuitive sense), it is controversial whether every feature of it is actually required by compositionality. All the features may, in sum, be sufficient but it is not clear that they are all necessary. With particular regard to the debate between DPL and DRT, the following two features are especially important. First, in Montague's system every syntactic object is assigned a denotation and, given a syntactic tree, every node can be assigned its denotation in a 'bottom-up' fashion. Simply assign to each leaf-node its denotation and then, for each sub-tree of depth 1 whose leaves have denotations assigned, calculate the denotation of the sub-tree's root by applying the semantic operation associated with the construction of the sub-tree to the denotations of the sub-tree's leaves. The roots of the sub-trees are then treated as leaves of other sub-trees in an iteration of the process. The procedure is repeated until one has calculated the denotation of the root of the whole tree.

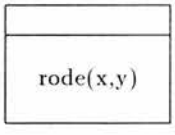
Secondly, Montague's demand that there be a homomorphism from the syntactic algebra into the semantic one constrains the use of any other object which might be used as an intermediary between syntax and semantics. Montague himself made use of the language of Intensional Logic (IL) which was neither part of the object language nor

contained in the semantic algebra. However, he was able to show that, despite his use of IL, there still *was* a homomorphism between the two relevant algebrae and therefore that his use of IL was not essential. The use of IL was not an ineliminable feature of his theory.

Neither of these two features of Montague's system appear to hold good of our exposition of DRT given in 2.1. The discussion of these two features forms the topic of the next two sections.

2.3.1 'Bottom-up' Compositionality

Zeevat in his 'Compositional Approach to DRT' ([Zeevat 89]) complains that the truth definition for DRSs is 'essentially a top-down interpretation'. It is perhaps not immediately clear what this means since 'top-down' and 'bottom-up' are predicates of *procedures* whereas what concerns Zeevat is a *definition*. Nevertheless, it is clear that our exposition of DRT is not immediately susceptible to the 'bottom-up' procedure for assigning denotations as described for Montague's system above. For instance, there are DRSs such as the following



which may appear within other DRSs and are not assigned an independent meaning by the rules for interpreting DRSs. The interpretation of DRSs does not consist of an algebra of syntax, an algebra of meanings and a homomorphism from the former to the latter. Zeevat therefore sets about rephrasing the definition in the style of Universal Grammar using a suitable syntactic algebra (of DRS-type objects) and a semantic algebra. A different and more explicitly top-down element of DRT is the DRS construction algorithm which recursively breaks down a complete syntactic structure for a sentence. Zeevat also re-writes the syntactic rules for English in a suitable manner, in order to 'obtain compositionality'.

It is worth pointing out that our exposition of DPL was *also* not compositional by this standard since neither was that presented in the style of Universal Grammar. Indeed Tarski's original truth definition for FOPL is not (strictly) compositional by this standard either. As Zeevat remarks, providing such an algebraic interpretation even for FOPL is a non-trivial task.

It is, however, most unclear why the top-down feature of the truth definition for DRSs should be taken as abandoning compositionality. The reason why it is unclear becomes apparent when one reflects on what it is one wants a compositional theory *for*. I do not deny that a bottom-up system may have many useful properties, *e.g.* in making proofs of properties of the system much easier to find and write down.

Compositionality is the semantic theorist's response to the fact that the expressions of human languages are built from finite and relatively small resources and yet there are an indefinitely large number of expressions and even indefinitely large expressions which we can understand. We can construct an expression which no-one has ever heard, uttered or thought before but which can nevertheless be understood. The reason is simply that the expression is constructed out of words which we have understood before put together into patterns which we have understood before. Old words in old patterns still yield new expressions. To capture this phenomenon, what we need to use is recursion; and characterizations of truth such as that used in the account of DPL (2.2) and that used by Tarski in his original account for FOPL are suitably recursive. The point was made several years ago by Davidson

> By giving such a theory, we demonstrate in a persuasive way that the language, though it consists in an indefinitely large number of sentences, can be comprehended by a creature with finite powers. A theory of truth may be said to supply an effective explanation of the semantic role of each significant expression in any of its appearances. Armed with the theory, we can always answer the question, 'What are these familiar words doing here?' by saying how they contribute to the truth conditions of the sentence. (*This is not to assign a 'meaning', much less a reference, to every significant expression*).
> [Davidson 68] (my italics)

Kamp has recently made similar comments in his discussion of DPL

> The ultimate motive ...[for compositionality] ...is, I take it, the insight that unless we provide a finite characterisation of the possible syntactic forms of a language as well as a finite characterisation of how each of these forms determines the meaning which it in fact carries, we have not explained how human languages can be learned or used. ...Compositionality, in the form in which Montague advocated it ...[seems] ...the most aesthetically satisfying form in which the learnability of a natural language could be explicated. But of course it is not the only way.
> [Kamp 90]

For these reasons, I shall not in this thesis be insisting on 'bottom-up' compositionality.

Groenendijk and Stokhof also suggest that DPL does not give you everything you might like in order to cope with compositionality 'below the sentential level'. In [Groenendijk & Stokhof 91a] they embark on the project of 'Dynamic Montague Grammar' (DMG) which contains, in particular, λ-abstraction. DMG can 'cope with the phenomena DPL deals with in a completely compositional fashion - below, on and beyond the sentential level'. But what they appear to mean by this concerns rather the translation relation between English sentences and DPL/DMG formulae, and not simply the truth definition of DPL. That is, it is one issue whether DPL itself has a compositional semantics and quite another whether we can use it in giving a compositional semantics for English. With regard to the latter, Groenendijk and Stokhof appear to be pointing out that although one can write down a DPL formula which translates 'Every man succeeded', namely $\forall x\ (m(x) \rightarrow s(x))$, one cannot write down a formula denoting what 'every' contributed to this translation. This is similar to the problem that Montague faced with FOPL and which he solved using λ-abstraction. It appears then that Groenendijk and Stokhof want a denotation for 'every' in English but not, say, for $\forall x$ in DPL. Do they not place a severer constraint on the truth definition for English than they do on their own formal language? Presumably they are just kind to their readers in not giving every definition in the style of Universal Grammar, though they undoubtedly could do so, if required.

Just as I do not insist on 'bottom-up' compositionality for dynamic logic I will also not insist that every syntactic unit of English should have some one object as its meaning (or some one formula as its translation) and so I shall not be concerned with the 'completely compositional' system of DMG. Indeed, in the next part of the thesis, I shall be concerned largely with ambiguous sentences and discourses where one cannot maintain the view that the sentences have some *one* meaning anyway. Montague's 'strong' system compels one to disambiguate ambiguous sentences in some way before interpreting - for example, by maintaining that only strings are ambiguous and syntactically structured sentences are not or that syntactically structured sentences are ambiguous but the analysis or derivation trees for them are not.

One of the contributions of this thesis will be to give a perfectly precise dynamic semantics for a fragment of English with quite ordinary surface syntactic structures. The fragment will be defined without using the resources of DMG. It will also be a fragment containing, as English contains, quantifier scope ambiguities and anaphoric ambiguities. That is, the input structures will not fully determine which quantifiers take relative scope over which other quantifiers, and nor will those structures be indexed indicating which pronouns depend on which antecedents for their interpretation. The issue of providing a compositional semantics for actual English, rather than the syntax of FOPL, is discussed further below.

2.3.2 Intermediate Semantic Levels of Representation

The second feature of Montague's system which does not hold true of our exposition of DRT is that it is not clear that the intermediate level of DRSs is, in principle, eliminable. It is this issue of the proper status of DRSs which has often been felt to be most objectionable. To be sure, proponents of DRT see it as a positive advantage, and Kamp claims in [Kamp 81] to be bringing together two different strands in semantics - one which 'sees meaning principally as that which determines conditions of truth' and one 'implicit in many studies by computer scientists (especially those involved with ar-

tificial intelligence), psychologists and linguists - studies which have been concerned to articulate the structure of the representations which speakers construct'. One primary question is whether the ineliminable intermediate representation (if it is ineliminable) actually forces one to adopt a psychological interpretation for it.

One popular line of thought connects this to our previous question concerning the desirability of a homomorphism between syntactic and semantic algebrae. Zeevat, for example, states that the relation between syntactic objects and their logical representations and between logical representations and their meanings must both be homomorphisms because 'In this way, the composition of both homomorphisms is itself a homomorphism between syntax and semantics. The level of representation has thereby only a secondary status in the theory ...one can in principle eliminate it'. If one cannot eliminate it, then a psychological interpretation may be encouraged. However, this seems rather strong given that the original Tarski truth definition is not presented as a homomorphism and no-one worries about the status of any intermediate representations in that definition.

A different way to appreciate the issue is to consider *use* versus *mention* of the meta-language in a semantic theory. In giving a semantic theory for a formalized language one *uses* a meta-language with at least the resources to refer both to expressions of the object language and to their denotations. Other resources will also be required in general in order to construct the rest of the semantic machine. However, one will not usually want to refer to syntactical properties of the meta-language. Otherwise it will look as if the semantical properties of the object language depend on the syntactical properties of the language in which the theory is stated. The meta-language must be used and not mentioned. It is for this reason that Montague demanded that the translation language IL be in principle eliminable. By showing that IL was only used to refer to objects in the model and that nothing depended on the particular syntax of IL, he thereby showed how IL could be thought of as just a part of the meta-language in a one-stage theory which did not use translations. The original Tarski truth definition

for FOPL is a simple one-stage theory. There is no 'intermediate' language but just an object language and a meta-language used to state its semantics.

Applying this lesson to the interpretation of DRSs, it follows one should not both appeal to syntactical properties of the language of DRSs (for example, the accessibility relation) and believe that that language is being *used* as part of the meta-language. If that language is referred to, then it should be conceived of as being a rather abstract level of syntactic structure of the object language. This is by no means an indefensible position ([van Eijck 85] defends it) though, by the standards of orthodox syntactic theorizing, it is an odd one. The usual forms of argument for constituency concerning distributional evidence, the possibility of co-ordination and the use of pro-forms, for example, do not seem to apply.

If DRSs *do* refer, *i.e.* if they are indeed used in the meta-language, then there is the issue of what they refer to. They certainly do not refer to objects in the model (since models are just pairs of a set of individuals and an interpretation function for some basic lexical items), though this is not the only option, as is sometimes implied. Assignment functions in first order logic are not *part* of the model either. According to Kamp, DRSs are, in a suitable sense, mental representations and not just technical devices used to oil the wheels of our semantic theorizing. DRT is intended to make definite psychological claims about the structure of our minds as we interpret sentences. This is quite different from most formal semantic theories. They take as their goal solely the prediction of truth conditions for sentences and their entailment relations. David Lewis, for example, asks whether a good grammar (including syntax, semantics and some pragmatics) *should* be suited to psycholinguistic theory but answers ' I doubt that it is worthwhile to pursue it in our present state of knowledge' [Lewis 81]. Kamp's opinion is that we will have to invoke the complexity of DRSs (for example, appeals to inferences that we humans actually make) eventually, in order to deal with linguistic data in all its richness, and that it makes sense now to start exploiting such a theory. In order to evaluate properly DRT's psychological claims, one would like really other

evidence apart from truth conditions for sentences. Otherwise, a theory which did not make psychological claims and could make the same truth conditional predictions would be preferable - the psychological claims of the rival appearing somewhat otiose.

I am not going to adjudicate here the difficult issue concerning the merits of starting to make our semantic theories psychologically testable at a finer granularity than just truth conditions. But part of the adjudication will turn on what can be done *without* invoking the complexity of DRSs, and this is sufficient to motivate the enterprise of dynamic logic with which this thesis is concerned. For the moment, it is sufficient to note that the semantic theories I will be presenting will not be invoking *any* intermediate language or representation, and therefore the question of its proper status or eliminability will simply not arise.

2.3.3 From Surface Structures to Meanings

It is easy to gain the impression from [Groenendijk & Stokhof 91b] that compositionality forms the *only* point of difference between DRT and DPL. DPL is designed (merely) to differ from other discourse theories in a methodological way, *i.e.* compositionally, and not empirically.

However, even a brief perusal of the two theories in sections 2.1 and 2.2 reveals a substantial difference in the scope of the two theories. Most obviously, DRT predicts truth conditions for English sentences as structured by a quite ordinary grammar. The input to the theory is natural language input. By contrast, DPL only defines a semantics for an already formalized language and it is not made explicit how natural languages are to be translated into DPL. Unless such a translation could be specified it would be quite unfair to claim that DPL was performing the same task as DRT, only it was doing so compositionally.

It is in the service of such a translation that DMG ([Groenendijk & Stokhof 91a]) is devised, as we noted earlier. That is, if one thinks of DPL as playing a role something like

that of IL in Montague's system, then we will need both to translate natural language into DPL (compositionally) and to interpret DPL in a model (compositionally).

It is one of the aims of this thesis to show how to give a compositional interpretation of English surface structures, without having to invoke a language such as DMG. It was noted earlier that DMG does not provide an explanation of how the two issues of relative quantifier scoping and pronominal resolution are to be dealt with. DRT, of course, does include a theory of pronominal resolution via the 'accessibility' condition on the rule **CR.PR**. A pronoun will be associated with a particular discourse referent introduced by some other noun phrase only if, at the point at which the pronoun is processed, that discourse referent is in an accessible position.

The compositional interpretation of surface structures will be given by a recursive truth definition for English. That is, whereas Groenendijk and Stokhof show how to interpret the syntax of FOPL dynamically, we shall provide a similar theory for English syntax. We will not be using Montague's route via a translation into a formal language whose interpretation we already have.

2.4 Summary

The fact that DRT is not immediately amenable to a 'bottom-up' procedure for assigning denotations I shall not consider to be a serious objection. The sense in which DRT departs from a standard semantics in its two-stage procedure with a psychological interpretation of the intermediate representative level is taken to be sufficiently serious to merit pursuit of dynamic logic. I shall also take it that a simple recursive truth-theory in the style of Tarski's semantics for FOPL meets, as Davidson suggests, the intuitive demands of compositionality and this is the style of semantic theory I shall be presenting throughout the thesis. I shall not be attempting to meet the overly strict demands of Universal Grammar. Nevertheless, since the goal is a compositional interpretation of natural language structures, we will also be specifying a truth theory

for such structures directly and not just for a formal language into which we must first translate.

Chapter 3

Quantifier Scoping Algorithms & Interpretations

3.1 Introduction [1]

Our objective in the first part of this thesis is to define a dynamic semantics for a language containing both quantifier scope ambiguities and ambiguities of pronominal resolution. This will be achieved without recourse to an intermediate disambiguated language. In this chapter, we concentrate on quantifier scope ambiguities and, since that phenomenon alone has no dynamic aspect, we do so in a traditional static framework.

It is important to treat quantifier scoping correctly first for two reasons. First, the issue of how to treat them inevitably raises once again the question of whether the treatment is compositional or not. If dynamic logic were compositional but the treatment of quantifier scoping were not, then the attractiveness of a dynamic interpretation of natural language containing quantifier scope ambiguities would be diminished. Secondly, it is important to treat quantifier scope ambiguities because of their possible interactions with anaphoric dependencies. Barwise has termed 7 a 'jungle path sentence' [Barwise 87].

(7) Statistics show that every 11 seconds a man is mugged here in N.Y. city. We are

[1] Part of the content of this chapter has appeared in [Lewin 90]

here to interview him

.

The idea is that if 'him' in the second sentence of the discourse is to be anaphorically dependent on 'a man' in the first sentence, then it must be the case that 'a man' takes wide scope with respect to 'every 11 seconds'.

The system of Dynamic Predicate Logic does not tell us how one is to decide which pronouns may anaphorically depend upon which antecedents. Examples such as 7 show us that, if syntax does not determine fully the relative scope of quantifiers then we cannot expect syntax to tell us fully which pronouns are anaphorically related to which antecedents. Co-indexing 'him' and 'a man' in 7 will still not mean that the two are anaphorically related. We could deal with this by disambiguating multiply quantified sentences syntactically (e.g. at a level of syntax called 'LF') - then a subsequent pronoun can anaphorically depend upon an earlier indefinite noun phrase if it is in a 'wide-scope' position in LF. However, in our dynamic environment, it is preferable to invoke the idea of allowing the pronoun to depend upon an antecedent which is *available at the time* where this latter class depends upon how other parts of the sentence were processed. This is the position I pursue in the next chapter. [Gawron & Peters 90] also contains arguments designed to show that a surface structure syntax (one not augmented with devices only present to help with the semantics) will not be able to determine all the various ways that sentences can be interpreted. Although their examples largely concern VP-anaphora, they also cite

(8) If every ballerina is too heavy for her partner, he will be displeased

The idea is that 'he' cannot anaphorically depend on 'her partner' if 'her' is bound by 'every ballerina'; whereas if we replace 'every' by 'a' then similar dependencies are acceptable. Since there is no difference between the two syntactic structures other than a lexical difference, syntax cannot determine the possible interpretations alone.

First, however, we need to account for quantifier scope ambiguities without appealing to a disambiguating level of syntax and without sacrificing compositionality.

3.2 Accounting for Quantifier Scope Ambiguities

The prima facie difficulty which quantifier scope ambiguities present for a compositional semantics is easily appreciated. If one takes to heart the slogan that 'the meaning of the whole is a function of the meaning of the parts and the way they are put together', then one immediately faces the problem that one cannot speak of *the* meaning of an ambiguous sentence. By definition, ambiguous sentences have more than one meaning. There are a variety of responses to this problem. One response is to claim, with [Montague 74b], that only strings of words are ambiguous but that syntactically structured sentences are not. Now one can speak of *the* meaning of *the* whole - namely the syntactically structured whole. Alternatively, one can claim, with [Cooper 83], that multiply quantified sentences are not syntactically ambiguous in virtue of being multiply quantified. Rather meanings are more complex entities and there are special interpretation rules of the form 'if P has meaning Q, then P also has meaning Q'' which allow multiple interpretations. One could now speak of the set of meanings of P. In a categorial framework, [Hendriks 90] has also advocated the use of sets of meanings where multiple members of a set derive from an initial member according to type-shifting rules. An interestingly different approach in the same framework is taken by [Emms 90] using polymorphic quantifiers.

In computational linguistics, the approach to quantifier scoping is sometimes seen as a two-stage approach. First, one constructs a syntactically unambiguous representation and, preferably compositionally, assign a meaning representation where relative quantifier scopes are undetermined. Secondly, one runs a special quantifier scoping algorithm on the meaning representation which generates all and only the correct scoped meanings. This approach has been advocated by, amongst others, [Schubert & Pelletier 82]

and [Alshawi *et al* 88]. One advantage claimed for it is that the means of quantifier scoping is thereby put into a separate module. This may not only be conceptually tidy but also allow the development of efficient algorithms for this phase of the work. However, one would not expect to insulate quantifier scoping entirely from other aspects of the grammar, especially pronominal resolution with which there are considerable interactions.

The two-stage approach is however vulnerable to the charge that it is not compositional. In particular, it is not clear that the intermediate meaning representations where quantifier scopes are not resolved are eliminable in the sense in which Montague's IL was. In the following sections I will first discuss Hobbs and Shieber's quantifier scoping algorithm [Hobbs & Shieber 87] which was proposed as a 'best yet' algorithm and has been incorporated into working systems, *e.g.* [Alshawi *et al* 88] and also a criticism of Pereira's that the algorithm is indeed non-compositional. After that, I shall put forward a semantic account of quantifier scoping and show how it naturally suggests an algorithm which represents an improvement on Hobbs and Shieber's.

The justification for one more approach to quantifier scoping - my own - is twofold. First, it is an extremely simple one requiring neither syntactic nor semantic complexity. The second justification is simply the light that it throws on Hobbs and Shieber's quantifier scoping algorithm - enabling us to see why that algorithm is successful, as well as suggesting an improvement. My account will not be 'bottom-up' compositional but will rely on a top-down interpretation procedure. The attractiveness of top-down interpretation procedures for dealing with quantifier scoping has been noted on several occasions previously, for example [Woods 68], [Hintikka 74], [van Eijck 85]. The account I will offer will differ from all of these however by providing a simple variant of the standard Tarski truth definition and not appealing to 'extended logical forms' (van Eijck), game-theoretical interpretations (Hintikka) or giving syntactical rules for manipulating formulae without a formal semantical interpretation (Woods).

3.3 Hobbs and Shieber's Quantifier Scoping Algorithm

The generation of unambiguous, formally specified and manipulable representations from English sentences has long been a goal of computational linguistics. The mechanisms for disambiguating relative quantifier scopings has also been discussed on several occasions (in particular, see [Woods 68], [Woods 78], [Schubert & Pelletier 82] as well as [Hobbs & Shieber 87]). In the latter paper, Hobbs and Shieber took as one of their goals simply the explicit statement of an algorithm for quantifier scoping, which was independent of any particular system or implementation. They also claimed their algorithm produced better results than previous attempts with respect to certain complex noun phrases.

The algorithm operates on an input representation defined as follows

> A well-formed formula (wff) in the input language is a predicate or other operator applied to one or more arguments. An argument can be a constant or variable, another wff, or what we will call a complex term. A complex term is an ordered triple consisting of a quantifier, a variable and a wff ... ([Hobbs & Shieber 87] p.49)

The example sentence 9a may therefore be coded as 9b in the input language.

(9) a some woman loves every man
b loves(<some x woman(x)>,<every y man(y)>)

The two noun phrases are represented as complex terms (distinguished by appearing within angle brackets) and left *in situ* as arguments to the verb. Scope dependencies are thereby left undetermined in the input language just as they are in surface structure English. The input language is meant to represent 'predicate-argument relations and the relations of grammatical subordination' (*op.cit* p.49). Complex terms can also be nested thereby enabling the representation of the complex noun phrases for whose treatment Hobbs and Shieber claim they have a superior algorithm. For example, the sentence 10a

(10) a some representative of every department arrived
b arrived(<some r rep-of(r,<every d dept(d)>)>)

The output representations generated by the algorithm are similar to the input representations except that there are no complex terms and expressions containing quantifiers are four part expressions consisting of a quantifier, variable and two formulae. So 9a is disambiguated by the production of the two formulae 11a,b and 10a is disambiguated by the production of 11c,d.

(11) a some(x,woman(x),every(y,man(y),loves(x,y)))
b every(y,man(y),some(x,woman(x),loves(x,y)))
c some(r,every(d,dept(d),rep-of(r,d)),arrived(r))
d every(d,dept(d),some(r,rep-of(r,d),arrived(r)))

The main operation employed in the algorithm, 'apply', takes a formula and a complex term occurring in the formula and generates a new formula not containing that complex term. If Σ represents a formula and $\Sigma(< quant\ var\ formula >)$ represents a formula containing a complex term, then 'apply' generates a four part formula whose first three parts are the parts of the complex term and whose fourth part is Σ with the variable replacing the complex term: $quant(var, formula, \Sigma[var/ < quant\ var\ formula >])$. Any application of 'apply' thus removes one complex term from an input formula. By recursive application, all complex terms are removed and the recursion terminates when there are no more complex terms in the formula. When a formula contains more than one complex term, there is a choice concerning which complex term to 'apply' first. Different choices generate different quantifier scopings.

If we consider 9b as an example, then choosing <every y man(y)> as the first complex term to be applied will result in the following representation

(12) every(y,man(y),loves(<some x woman(x)>,y))

Subsequent application of <some x woman(x)> results in

(13) some(x,woman(x),every(y,man(y),loves(x,y)))

which is the disambiguation stated in 11a. Complex terms which are chosen for application *earlier* thereby receive *narrower* scope than those chosen later.

The algorithm presented in [Hobbs & Shieber 87] can be considered to be an augmented version of the basic algorithm presented above. One augmentation concerns the handling of so-called *opaque* arguments of higher order predicates. This naturally brings to mind an operator such as 'believes' which also interacts with quantifier scopes. Thus 'Jack believes every man is mortal' can arguably be held to be true either if Jack has the general belief that each and every man is mortal or if it is true that for each man that Jack believes that that man is happy even though he does not have the general belief itself. I shall not concern myself with this particular augmentation here since I am not considering issues of intensionality. It is worth pointing out however that the sense of 'opacity' made use of by the algorithm is somewhat non-standard. For example, Hobbs and Shieber treat negation as an opaque predicate because quantifiers can also take scope relative to negation (witness the ambiguity in 'Everyone isn't here'). The standard use of 'opaque' concerns whether a given context allows substitution of co-referring names *salva veritate*, whereas negated contexts do not usually present this difficulty (though there are problems if empty or non-denoting names are allowed). Even Quine doesn't object to quantification into a negated context.

Hobbs and Shieber's method does not anyway extend to intensional verbs such as 'seeks'. Montague, famously, treats the ambiguity involved in 'John seeks a unicorn' whereby, it is claimed, either there may be some particular unicorn which John seeks or else there are, in fact, no unicorns but John still seeks one. This cannot be handled by Hobbs and Shieber's algorithm because the basic operation 'apply' always takes a complex term and gives the associated quantifier scope over the formula in which it appears - so 'a unicorn' is bound to take scope over 'seeks'. Montague's solution is obtained precisely by not using a quantifier raising or extraction rule, but by giving a denotation to 'a unicorn' (namely, a function from possible worlds to the set of all sets containing an individual which is a unicorn in that possible world) and leaving this

denotation as an argument to the relation denoted by 'seeks'.

Although I shall not be concerned with opaque predicates in general, I will discuss how to handle negation and other similar 'higher-order predicates' such as conjunction, at the end of this chapter.

There are two other augmentations to the basic algorithm. The first can be motivated by consideration of complex noun phrases as represented in 10b. If we choose to apply the most deeply nested complex term first <every d dept(d)>, then the result is

every(d,dept(d),arrived(<some r rep-of(r, d))>))

and subsequent application of the remaining complex term results in

some(r, rep-of(r, d),every(d,dept(d,arrived(r))))

which is not a reading of the input sentence. The representation contains a free variable d which is intended to be bound by the universal quantifier although it is not, in fact. In order to prevent this, Hobbs and Shieber forbid the algorithm to apply complex terms which are embedded within other complex terms. However, such a prohibition now means that the sentence is predicted not to be ambiguous since there is now only one order in which the complex terms can be applied. The outermost complex term must be applied first and then the nested complex term. Yet the sentence does appear to be scopally ambiguous. Hobbs and Shieber's solution is to add an extra step to the algorithm whereby, instead of simply copying the third part of the complex term (the 'restriction') into the resulting output representation when we apply the complex term, we may first recursively apply complex terms within the restriction. Therefore, when we apply <some r rep-of(r, <every d dept(d)>)> to its containing formula, there are two possible results. If we choose the extra step of scoping the restriction then the complex term embedded in it will receive narrow scope with respect to the applied complex term. If we do not choose this extra step, then the result will be a formula still containing the nested complex term <every d dept(d)> which, by a recursive call

of 'apply', will receive wide scope.

In order to deal with complex noun phrases then, the general policy underlying the basic algorithm that complex terms chosen first receive narrow scope is abandoned so that we can generate alternative readings for sentences involving complex noun phrases.

The second modification results from the observation that the representation of 14a given in 14b will still be converted into a formula containing a free variable

(14) a Every man saw a picture of himself
b saw(<every x man(x)>,<a y picture(y,x)>)

Suppose we apply <every x man(x)> first, then the result will be

(every x man(x) saw(x,<a y picture(y,x)>))

Application of <a y picture(y,x)> will now lead to x being free. Hobbs and Shieber prevent this by stipulating that a complex term is applicable only if all free variables in the complex term to be applied are also free in the containing formula. This prevents application of <a y picture(y,x)> because x is free in the term but not free in the containing formula.

Is The Algorithm Compositional?

Both modifications to the basic algorithm have been motivated by the need to prevent the appearance of 'free variables' in the output representations. Pereira has dubbed this the 'Free Variable Constraint' [Pereira 90]. As we have seen, one mechanism adopted to meet the constraint is to allow the algorithm to keep track of which variables are free and which are not free in the representations it modifies. Pereira claims that this mechanism is objectionable for two reasons. First, the mechanism requires a logical-form level of representation. Secondly, the mechanism is not compositional because the constraint (that variables free in the complex term must be free in the containing formula) is stated in terms of syntactic properties of the formulae used to denote

meanings and not the meanings themselves. Pereira does not state precisely why his first reason constitutes an objection and restricts himself to the observation that the role of logical form in linguistic theory depends on 'non-trivial assumptions'. We might note that the objection cannot be to using logical forms to *filter* out unwanted representations since Hobbs and Shieber's algorithm does not function like that. In fact, they prove that the algorithm never produces a formula containing a free variable.

Pereira's second objection clearly relates to the point raised in chapter 2 that in stating a semantical theory in a metalanguage, one does not want to *mention* that metalanguage, but only use it. Montague proved that he only used IL to denote meanings in the model and nothing hinged on the syntactic peculiarities of IL itself. Does Hobbs and Shieber's algorithm fall foul to this objection? It only makes sense to complain about reference to syntactic properties of expressions used to denote meanings if those expressions are given denotations in the first place. Yet nothing in our exposition of the algorithm so far has concerned assigning meanings to either the representations or the operations the algorithm employs. Hobbs and Shieber confine themselves to the observation they do not provide a semantics for the input language and that 'the output language has, of course, a standard logical semantics' (*op.cit.* p.60). The question is whether one can *only* view Hobbs and Shieber's algorithm as an algorithm that does a job - generating one set of representations from another, where the latter can subsequently be model-theoretically interpreted. A more attractive perspective is one which explains why the algorithm is a good one by relating it to a semantical theory which explains the properties of wholes in terms of those of its parts. Pereira's second objection is best understood as the claim that such a relation will be hard to find because the algorithm refers to logical form syntax and that is just what a compositional semantic theory should not do.

The semantical theory to be given in section 3.4 will relate quite naturally to a quantifier scoping algorithm, in fact, to one which bears a very close relation, though not identity, to Hobbs and Shieber's algorithm. In this way, the success of the algorithm,

which also improves on Hobbs and Shieber's, is explained.

3.3.1 Interpreting the Algorithm

In this section, I further discuss the issue of interpreting the workings of the algorithm by considering the concept of 'soundness'.

One of Hobbs and Shieber's claimed advantages of their algorithm is that it is less profligate than many other accounts in the readings generated for certain complex noun phrases (here they cite [Montague 74a] and [Cooper 83]). Another advantage is the proofs provided of certain properties of the algorithm, *e.g.* termination. One property they do not prove is 'correctness in the sense of showing that the algorithm is semantically sound, *i.e.* that it yields wffs with interpretations consistent with the interpretation of the input expression'. They do not prove this 'simply because we do not provide a semantics for the input language'. Neither do they prove completeness 'depending as it does on just which scopings one deems possible'. If by 'input expressions' is meant simply the *initial* representations fed into the algorithm, then this lacuna can be easily put right. We can provide a semantics for the input expressions by adopting, for example, a formal-semantical theory along the lines of [Montague 74a] and [Cooper 83]. Unfortunately, the algorithm will be incomplete with respect to these accounts, given Hobbs and Shieber's claim that the algorithm is less profligate than they are. In this respect it is difficult for Hobbs and Shieber to claim superiority for their algorithm since the objectives of the two clearly differ. The accounts of Montague and Cooper aim at a formal semantical account of English syntactic structures whereas Hobbs and Shieber's aim is, on this interpretation, to provide an algorithm which will produce the *same results* as a correct formal semantical theory. From this perspective, Hobbs and Shieber (and Pereira) need not worry themselves over any lack of compositionality - for to worry about that is precisely to worry about *how* the output meanings are produced on the basis of the input structures and their meanings, whereas all that is currently required is an algorithm with the right input-output patterns.

An alternative way to view the algorithm is as *stipulating* the meanings of the input expressions. Under this interpretation there is no need to provide an alternative formal semantics with respect to which one can check whether the algorithm is sound and complete. One simply needs to check that the meanings that the algorithm provides for the input sentences are those that English speakers actually assign to those sentences. However, the compositional issue now re-emerges in the question of how the statements of the truth conditions of sentences emerges from *semantic* properties of the parts that make them up. Nothing has been said about how sub-sentential parts have a semantic function other than that an algorithm manipulates pieces of syntax into formulae which do state the truth conditions of the input sentence.

Could this deficiency not be remedied ? I think there is a difficult problem here for Hobbs and Shieber because the algorithm maintains no clear distinction between object language and meta-language. Clearly the initial input representations can be thought of as object language structures and the final output representations can be seen as meta-language statements of the truth conditions of the object language, but the intermediate structures over which the algorithm recurses are neither one nor the other. For example, the intermediate structure we constructed above (12 repeated below as 15) neither represents 'predicate-argument relations and the relations of grammatical subordination' in English nor is it part of the output language because it contains a complex term.

(15) every(y,man(y),loves(<some x woman(x)>,y))

Such formulae cannot be seen as part of the object language or part of the meta-language.

We have seen that Pereira objected to appeals made to logical form syntax. Since one must check the free variables of a formula such as 15, he must consider that 15 is part of logical form. It then follows directly that the recursive account of the truth conditions of object language sentences proceeds not by recursion over object language

structures but by recursion over logical forms. It is this fact which is objectionable if we are trying to understand how object language structures contribute to meanings, and not just an apparently minor point about logical form free variables.

By contrast, the structures over which our algorithm will operate will be object language structures only. The difference can be compared to that between iteration and recursion - rather than iteratively *modifying* a representation, we will recursively descend the input structure and build the output structure as we successively exit each level of recursion. At no point will a piece of built structure determine how the algorithm proceeds. The function of the output structures will only be to state the truth conditions of the input structures.

3.4 An Alternative Account

The account of quantifier scoping I want to offer is a direct and recursive account of the notion of 'satisfaction of ϕ', where ϕ ranges over English syntactic structures. This procedure is, of course, the one first used by Tarski in his seminal paper [Tarski 56], and in DPL. The difference is that those accounts were for fully disambiguated languages whereas we shall now characterize satisfaction, directly, for an ambiguous language. The compositional demand will again be met by using a recursive account of satisfaction.

There are two ways, in general, to give a recursive definition. One is to list the base cases and then state the inductive cases using biconditionals. This is the method Tarski adopted. An alternative method is to list the base cases as before but then state the inductive cases using simple conditionals rather than biconditionals. In the second case, it is then necessary to add a *general exclusion clause* with the purport that nothing falls under the concept being characterized unless it does so in virtue of the conditional statements just given. Usually nothing much depends on which method is used. In the case of quantifier scope ambiguity however, I suggest the latter method has

a distinct advantage. The reason is that if our theory contains several conditionals and an exclusion clause it is possible to use more than one route through the conditionals in order to arrive at a given result. For example, we can give the following inductive definition of a propositional language which is ambiguous with respect to the scope of $\wedge$ and $\vee$.

Basis	Let p, q, r be atomic sentences
Induction	1. If α and β are sentences, then so is $\alpha \wedge \beta$
	2. If α and β are sentences, then so is $\alpha \vee \beta$
Exclusion	Nothing is a sentence except in virtue of the above clause

It is easy to verify that $p \wedge q \vee r$ is a sentence according to the above definition. Furthermore, there are two distinct proofs that it is a sentence. One involves showing that p is a sentence and that $q \vee r$ is a sentence. The other involves showing that $p \wedge q$ is a sentence and that r is a sentence.

The contrast between the two types of inductive definition is directly related to that between an algebra and a 'free' algebra, as used, for example, by [Montague 74b]. Montague's definition of a disambiguated language is set up precisely so that the result is a free algebra - that is, that no expression can be produced by more than one rule (technically, structural operation) or by the same rule with different arguments, and that no basic expression is also produced by a rule. I discuss the relation between my account and Montague's further below.

The general strategy for handling quantifier scope ambiguities is therefore to use an inductive definition using only conditionals and not biconditionals, together with a general exclusion clause. This gives us the freedom to take more than one route through the definitions and therefore generate the appropriate ambiguities. However, this freedom I will invoke in the semantic theory itself - not just in the definition of well-formed sentences, as was the case in our propositional example above.

3.4.1 Ambiguous Truth Theory

I now present an ambiguous truth theory for Hobbs and Shieber's input language, which I shall dub 'HSL'. Subsequently, I shall define such a theory over a fragment of English.

HSL syntax

First, we define a set of *pre-formulae* for HSL and then we specify a subset of this set which contains HSL formulae.

1 There is a basis set containing individual constants, n-ary predicates and an infinite number of variables.
2 If $t_1, \ldots, t_n$ are individual constants, variables or complex terms and P is an n-ary predicate, then $P(t_1, \ldots, t_n)$ is a pre-formula.
3 If ϕ is a pre-formula, and x is a variable, then $< \exists\ x\ \phi >, < \forall\ x\ \phi >$ and $< Most\ x\ \phi >$ are complex terms.
4 Nothing is a pre-formula or complex term except in virtue of the above clauses.

An HSL formula is defined to be any HSL pre-formula which contains no two complex terms that bind the same variable. For example, $loves(< \exists\ x\ man(x) >, < \forall\ x\ woman(x) >)$ is a pre-formula which is *not* also a formula whereas $loves(< \exists\ x\ man(x) >, < \forall\ y\ woman(y) >)$ is both a pre-formula and a formula. Complex terms are thereby uniquely indexed within a formula by the variables that they bind. (This assumption is also built into Hobbs and Shieber's algorithm although it is not mentioned in the definition of their input language).

If Σ is a formula containing a particular complex term π, then we may also denote this by $\Sigma(\pi)$. $\Sigma[\psi/\pi]$ denotes the very same formula as Σ except that the complex term π will have been replaced with ψ.

HSL semantics

A model for HSL is a pair $< D, F >$ where D is a non-empty set of individuals and F is a function from individual constants to members of D, and from n-ary predicates to n-ary relations on D. Assignments are partial functions from variables to members

of D.

For individual constants and variables, $[\![x]\!]^g$ is $g(x)$ for all variables x ; $[\![c]\!]^g$ is $F(c)$ for all constants c.

1 $[\![P(t_1,\ldots,t_n)]\!]^g = 1$ if $< [\![t_1]\!]^g,\ldots,[\![t_n]\!]^g > \in F(P)$
(where t_i are variables or individual constants)

2 $[\![\Sigma(< \exists\ x\ \phi >)]\!]^g = 1$ if $\exists\alpha\ ([\![\phi]\!]^{g^\alpha_x} = 1; [\![\Sigma[x/ < \exists\ x\ \phi >]]\!]^{g^\alpha_x} = 1)$

3 $[\![\Sigma(< \forall\ x\ \phi >)]\!]^g = 1$ if $\forall\alpha\ ([\![\phi]\!]^{g^\alpha_x} = 1; [\![\Sigma[x/ < \forall\ x\ \phi >]]\!]^{g^\alpha_x} = 1)$

4 $[\![\Sigma(< Most\ x\ \phi >)]\!]^g = 1$ if $\text{Most}\alpha([\![\phi]\!]^{g^\alpha_x} = 1; [\![\Sigma[x/ < Most\ x\ \phi >]]\!]^{g^\alpha_x} = 1)$

Ex. No formula is satisfied by an assignment g except in virtue of the above rules.

An HSL-formula ϕ is true just in case $[\![\phi]\!]^\emptyset = 1$.

3.4.2 Necessary and Sufficient Conditions

The theory outlined above entails that we must be a little more careful than usual in stating the relation between meanings and truth conditions. The standard account is well-represented by [Dowty *et al* 81] who state that 'to give the meaning of a sentence is to specify its truth conditions, *i.e.*, to give necessary and sufficient conditions for the truth of that sentence' (p.4). I do not wish to consider the general (philosophical) issue of precisely what content to ascribe to this equation. It is, however, worth pointing out that the 'ambiguous truth theory' does not immediately derive a *necessary* condition for the truth of 'loves($< \exists\ x$ woman(x) $>$,$< \forall\ y$ man(y) $>$)'. By working our way through the conditionals, we only derive two sufficient conditions.

(16) a loves($< \exists\ x$ woman(x) $>$,$< \forall\ y$ man(y) $>$) is true if
$\exists\ \alpha(woman(\alpha); \forall\ \beta(man(\beta); loves(\alpha,\beta)))$

b loves($< \exists\ x$ woman(x) $>$,$< \forall\ y$ man(y) $>$) is true if
$\forall\ \beta(man(\beta); \exists\ \alpha(woman(\alpha); loves(\alpha,\beta)))$

Once we have these two results and shown that these are the only results, then it is not difficult to derive additionally

(17) a loves($< \exists\ x$ woman$(x) >, < \forall\ y$ man$(y) >$) is true iff
$\exists\ \alpha(woman(\alpha); \forall\ \beta(man(\beta); loves(\alpha, \beta)))\ \vee$
$\forall\ \beta(man(\beta); \exists\ \alpha(woman(\alpha); loves(\alpha, \beta)))$

Is it the necessary and sufficient condition that gives the meaning of the sentence here ? It is much more natural to suppose that each of the two disjuncts 'gives the meanings' of the input sentence than that the disjunction itself 'gives the meaning'. The reason for this is that if we suppose that the whole disjunction gives the meaning then we are bound to ask what part of the object language structure gives rise to the disjunctive connective ? Yet there is no part of the input structure which is responsible for this - it is simply that the input structure *underdetermines* how to proceed through the semantic theory.

The ambiguous truth theory I have given seems to me to place quantifier scope ambiguities in precisely the right light. The compositional aim is met by recursion over object language syntactic structures, but recursion over the structure itself only gives rise to *two disjuncts*. Once we see that our theory produces two disjuncts we can then make a further inference. We can infer (by a rule of disjunction introduction) that the object language sentence in question is true if either of the two disjuncts is true - but this extra inference step is not licensed by any part of the syntax of the input sentence, only by our knowledge of the meta-language in which we state our semantic theory. Presumably it is possible someone might know that 'some woman loves every man' is true in one type of situation and know that it is true in another without ever realising that the sentence is, in fact, ambiguous. Arguably, this is true of most people who have never studied semantics. It is also worth pointing out here that the generation of the two disjuncts is not produced by supposing that there is 'really' more than one syntactic structure for the input sentence. I will discuss this point more fully when I come to compare the theory with other quantifier scoping theories.

3.5 A Modified Quantifier Scoping Algorithm

The theory stated above naturally suggests a (non-deterministic) algorithm for generating quantifier scope ambiguities.

First, we provide some explanation of the lisp-like language used to state the algorithm, in the style of [Hobbs & Shieber 87]. The expression '**let** *assignments* **in** *body*' means local variable assignment for the scope of *body*, where the value of the expression as a whole is the value of *body*. An assignment takes the form '*pattern* := *expression*' where *pattern* is a structure that pattern matches with the value of *expression* assigning any variable names in *pattern* to corresponding parts of the value of *expression*. Also, '**function** *function-name*(arg_1,...,arg_n); *body*' evaluates to an n-ary function whose name is *function-name*, whose arguments are named by arg_i and whose body is an expression. To evaluate a function, first one evaluates the arguments and then one evaluates the body. The value of the body is returned as the value of the function. Naturally, '**if** *expression-1* **then** *expression-2* **else** *expression-3*' returns the value of *expression-2* if the value of *expression-1* is **true**, otherwise it returns the value of *expression-3*. Any other expression evaluates to itself.

The following four functions are used in the algorithm: *term* is a unary function that takes an HSL formula as argument and returns **true** if there is a complex term in the argument (otherwise **false**); *applicable-term* is a unary function that takes an HSL formula as argument and returns, non-deterministically, a complex-term occurring in that formula; *subst* is a ternary function that takes a variable, a complex-term and a formula and returns the formula except that the complex-term is replaced by the variable; and *wff* is a quaternary function which evaluates four expressions q, v, $e1$, $e2$ and constructs a complex output language expression out of their values q', v', $e1'$, $e2'$. The expression '$q'(v', e1', e2')$' is returned as the value of *wff*.

The main function is *apply-terms* whose argument is an HSL formula and which non-deterministically returns a fully scoped version of the HSL formula in the output lan-

guage.

```
function apply-terms(form);
  if      term(form)
  then    let < quant var restrict> := applicable-term(form)
          in wff(quant,
                 var,
                 apply-terms(restrict),
                 apply-terms(subst(var,< quant var restrict>,form)))
  else    form
```

A program implementing the algorithm can be found in Appendix A.1.

The critical step of the modified algorithm is the following: given an input expression containing a complex term $\Sigma(< quant\ var\ formula >)$ construct a four part output expression whose first two parts are *quant* and *var* and whose latter two parts are the results of recursing on *formula* and $\Sigma[var/ < quant\ var\ formula >]$. The process is therefore very similar to that adopted by Hobbs and Shieber, but the difference is that our algorithm is working its way through the input *outside-in*. Quantifiers which are chosen earlier for application receive *wider* scope than those chosen later. This contrasts with the basic step of Hobbs and Shieber's algorithm.

This small difference has interesting consequences. First, consider the treatment of complex noun phrases (complex terms that contain other complex terms). It is precisely the treatment of these expressions for which Hobbs and Shieber claim their algorithm constitutes a significant improvement (though they also acknowledge the successful treatment in [Keller 86]). Having outlined the basic step of our modified algorithm, the treatment of nested complex terms follows without the need for any subsequent modification. Consider again 10a,b repeated below as 18a,b

(18) a some representative of every department arrived
b arrived(<some r rep-of(r, <every d dept(d) >) >)

We will apply *apply-terms* to 18b. Clearly, the antecedent of the conditional *term* will succeed. Suppose *applicable-term* returns <every d dept(d) >. Then *wff* will

return a statement made from 'every',d, and the results of recursions on 'dept(d)' and 'arrived(<some r rep-of(r, d) >)'. The first recursion will simply return what is supplied (since there are no complex terms in it and so *term* will return *false*). On the second recursion *applicable-term* will return <some r rep-of(r, d) >. *wff* will generate another four-part construction. It will consist of 'some', r and the results of recursions on 'rep-of(r, d)' and 'arrived(r)', *i.e.* some(r,rep-of(r, d),arrived(r)). The result of our initial call to *wff* will therefore be: every(d,dept(d),some(r,rep-of(r, d),arrived(r))). It is easy to check that had *applicable-term* returned the two complex terms in reverse order, the result would have been: some(r,every(d,dept(d),rep-of(r, d)),arrived(r)). Therefore we have generated the two disambiguations suggested earlier in 11c,d.

The net result is that our simple modification to the basic algorithm results in the ability to handle more complex cases (nested complex terms) without the need for any additional machinery. In contrast, the basic step in Hobbs and Shieber's algorithm was not in itself sufficient to deal with these more complex cases.

Furthermore, the new algorithm has a very natural interpretation. The function *apply-terms* takes object language syntax for its argument and takes one step through the associated truth theory. The outputs of the algorithm consist solely of meta-language expressions used to denote the values of the input object language expressions. Given an input of '$P(t_1, \ldots, t_n)$' (where none of the t_i are complex terms), *apply-terms* will return '$P(t_1, \ldots, t_n)$', just as it does in Hobbs and Shieber's algorithm. However, the important point is that the returned value is now to be interpreted as a meta-language statement of the satisfaction conditions of the input formula. To make things clearer, we could properly distinguish between object and meta-language expressions, for example by underlining all meta-language constructs. If we did that then *apply-terms* would return '$\underline{P(t_1, \ldots, t_n)}$' when presented with '$P(t_1, \ldots, t_n)$'. This corresponds to an application of rule 1 from the semantic theory 3.4.1. Similarly, given an input of 'arrived(< $\forall\ x\ man(x)$ >)', *apply-terms* would now return '$\forall\ x\ \underline{(man(x); arrived(x))}$' and it would arrive at such results by recursive calls to *apply-terms* with arguments

'$man(x)$' and '$arrived(x)$'. This corresponds to an application of rule 3. At no point will an underlined formula become an *argument* to the function *apply-terms* - which is as it should be if *apply-terms* is to be a relation between object language syntax and meta-language statements of satisfaction conditions.

Let us underline the point by rewriting the algorithm like so

function *apply-terms(form);*
  if *term(form)*
  then **let** < *quant var restrict*> := *applicable-term(form)*
    in *wff(*<u>*quant*</u>*,*
      <u>*var*</u>*,*
      apply-terms(restrict),
      apply-terms(subst(var,< *quant var restrict*>*,form)))*
  else <u>*form*</u>

By inspection, we see that underlined expressions are always returned only as *apply-terms* exits and never become the input to another function. Therefore, the algorithm never makes an appeal to the 'syntax of logical form' and we need not fear that it is non-compositional.

3.6 Ambiguity for an English Fragment

I have described a theoretical basis for dealing with languages containing quantified nounphrases (or 'complex terms') where the scope of the quantifiers is not determined by syntactical features of the object language. The example I have used is a version of Hobbs and Shieber's input language which is intended to reflect aspects of English syntax ('predicate-argument relations and the relations of grammatical subordination'). In this section, I use the same theoretical basis for a minimal English syntactic fragment based on the one stated in [Rooth 87]. In the next chapter, I will be concerned with anaphora and, in particular, with donkey anaphora so it makes good sense to use a fragment such as Rooth's which was itself proposed as a reasonable (if minimal) fragment in which to test and discuss the semantics of quantification and anaphora.

RL (Rooth Language) Syntax

The syntax is identical to the minimal fragment appearing in [Rooth 87] except that *RL* employs a different sort of index from Rooth's original fragment. Rooth's indices are assigned to noun phrases for the purpose of 'indicating co"reference" '. That is, the co-indexing of distinct noun phrases represents the result of a decision that there is an anaphoric dependency between them. We are not concerned with anaphora in this chapter so the indices in *RL* do not represent such a decision. The indices in *RL* are simply used to record, within a complex noun phrase, which noun phrase it is. So if 'man with a donkey' is a part of 'every man with a donkey' then we will mark the former phrase with the index of the latter. In this way, when we evaluate 'man with a donkey' we will know which object 'man' is being predicated of and which object stands in the 'with' relation to some donkey. Each noun phrase in the surface string will have a unique index, corresponding to the fact that in *HSL* each complex term was uniquely indexed by the variable it bound.

RL Rules	NP	→ DET N			
	N	→ N PP		PP	→ P NP
	S	→ NP VP		VP	→ V NP
	D	→ S D		D	→ S

RL Lexicon	NP	→ it
	DET	→ every|a
	N	→ man|woman|donkey
	P	→ with
	VP	→ runs| sat down| overate
	V	→ loves|feeds

Indexing Rules

1 Assign each NP a unique index
2 Assign any N immediately dominated by an indexed node that same index.

Indices will appear as superscripts to the node name to which they are attached. For ease of reading, where α is a lexical entry I write α_β instead of $[_\beta\ \alpha]$ and I shall also use Λ_β in general to mean the tree whose root is labelled with β. So, every$_{DET}$ means $[_{DET}$ every] and Λ_{NP^5} means the tree whose root is labelled with NP^5. As a larger example, 19a receives a structure shown in 19b, which we can also write as 19c.

(19) a Every man with a donkey loves a woman

b $[_S\ [_{NP^1}\ \text{every}_{DET}\ [_{N^1}\ \text{man}_{N^1}\ [_{PP}\ \text{with}_P\ [_{NP^2}\ \text{a}_{DET}\ \text{donkey}_{N^2}]]]$
$[_{VP}\ \text{loves}_V\ [_{NP^3}\ \text{a}_{DET}\ \text{woman}_{N^3}]]]$

c $[_S\ \Lambda_{NP^1}$
$[_{VP}\ \text{loves}_V\ \Lambda_{NP^3}]]$

RL Semantics

A model for *RL* is a pair $< D, F >$ where D is a non-empty set of individuals and F is a function from lexical N and VP to subsets of D, from lexical V to binary relations on D, and from lexical P to functions from members of D to subsets of D. Assignments are partial functions from indices to members of D.

Quantifier rules

1. $[\![\ \Sigma([_{NP^i}\ \text{every}_{DET}\ \beta_{N^i}])\ ]\!]^g = 1$ if $\forall x\ ([\![\ \beta_{N^i}\ ]\!]^{g_i^x} = 1\ ;\ [\![\ \Sigma[it_{NP^i}/\Lambda_{NP^i}]\ ]\!]^{g_i^x} = 1)$

2. $[\![\ \Sigma([_{NP^i}\ \text{a}_{DET}\ \beta_{N^i}])\ ]\!]^g = 1$ if $\exists x\ ([\![\ \beta_{N^i}\ ]\!]^{g_i^x} = 1\ ;\ [\![\ \Sigma[it_{NP^i}/\Lambda_{NP^i}]\ ]\!]^{g_i^x} = 1)$

Structural rules

3. $[\![\ [_S\ [_{NP^i}\ \text{it}]\ \beta_{VP}]\ ]\!]^g = 1$ if $[\![\ \beta_{VP}\ ]\!]^g(g(i))$

4. $[\![\ [_{N^i}\ \alpha_{N^i}\ \beta_{PP}]\ ]\!]^g = 1$ if $[\![\ \alpha_{N^i}\ ]\!]^g = 1\ \wedge\ [\![\ \beta_{PP}\ ]\!]^g(g(i))$

5. $[\![\ [_{PP}\ \alpha_P\ [_{NP^i}\ it]]\ ]\!]^g =$ $F(\alpha_P)(g(i))$

6. $[\![\ [_{VP}\ \alpha_V\ [_{NP^i}\ it]]\ ]\!]^g =$ $F(\alpha_V)(g(i))$

Lexical Insertion Rules

7. $[\![\ \alpha_{VP}\ ]\!]^g =$ $F(\alpha)$

8. $[\![\ \alpha_{N^i}\ ]\!]^g =$ $F(\alpha)(g(i))$

Discourse Rules

9. $[\![\ [_D\ \alpha_S\ \beta_D]\ ]\!]^g = 1$ if $[\![\ \alpha_S\ ]\!]^g = 1\ \wedge\ [\![\ \beta_D\ ]\!]^g = 1$

10. $[\![\ [_D\ \alpha_S]\ ]\!]^g = 1$ if $[\![\ \alpha_S\ ]\!]^g = 1$

Ex. No formula is assigned a semantic value except in virtue of the above rules.

A formula ϕ is *true* just in case $[\![\ \phi\ ]\!]^{\emptyset} = 1$.

As in our earlier rules for HSL, the quantified noun phrases are handled by an *outside-in* rule which effectively searches within a syntactic structure for an occurrence of a quantified noun phrase. Having found one, the rule then recurses first on the restriction

attached to the quantifier and then on the whole structure where the noun phrase is replaced by a pronoun. Since all occurrences of quantifiers are handled by these rules, all the other rules need only refer to structures which contain pronouns. So, while it might at first sight seem overly restrictive that, *e.g.*, rule 6 only deals with VPs that contain a pronoun, this is not the case because other noun phrases are dealt with elsewhere. The overall strategy is further discussed below (section 3.7.1).

3.6.1 Example

An example derivation for *RL* is now presented to illustrate the system at work. Consider again our earlier example repeated below.

(20) a Every man with a donkey loves a woman
b $[_S \quad [_{NP^1}\ \text{every}_{DET}\ [_{N^1}\ \text{man}_{N^1}\ [_{PP}\ \text{with}_P\ [_{NP^2}\ \text{a}_{DET}\ \text{donkey}_{N^2}]]]$
$[_{VP}\ \text{loves}_V\ [_{NP^3}\ \text{a}_{DET}\ \text{woman}_{N^3}]]]$

The sentence is true just in case it is satisfied by $\emptyset$. In the following derivation, I apply a number of rules in parallel in one step. The rules applied are given in a list, *e.g.* [1,2,3], after the meta-language statement they are applied to and above the meta-language statement that they generate. The first rule in the list is applied to the first line of the previous statement and generates the first line of the next, the second applies to the second line of the previous statment and so on. The rule number '-' indicates a simple unnumbered reduction, *e.g.* from $\emptyset^x_1(1)$ to x, or else no reduction at all.

$[\![$ Every man with a donkey loves a woman $]\!]^{\emptyset} = 1$ iff ...

[1]

$\forall x$ ($[\![[_{N^1}\ \text{man}_{N^1}\ [_{PP}\ \text{with}_P\ [_{NP^2}\text{a}_{DET}\ \text{donkey}_{N^2}]]]]\!]^{\emptyset^x_1} = 1$;
$[\![[_S\ \text{it}_{NP^1}\ [_{VP}\ \text{loves}_V\ [_{NP^3}\text{a}_{DET}\ \text{woman}_{N^3}]]]]\!]^{\emptyset^x_1} = 1$)

[2,2]

$\forall x$ ($\exists y$ ($[\![\text{donkey}_{N^2}]\!]^{\emptyset^{xy}_{12}} = 1$;
$[\![[_{N^1}\ \text{man}_{N^1}\ [_{PP}\ \text{with}_P\ \text{it}_{NP^2}]]]\!]^{\emptyset^{xy}_{12}} = 1$)
$\exists z$ ($[\![\text{woman}_{N^3}]\!]^{\emptyset^{xz}_{13}} = 1$;
$[\![[_S\ \text{it}_{NP^1} [_{VP}\ \text{loves}_V\ \text{it}_{NP^3}]]]\!]^{\emptyset^{xz}_{13}} = 1$))

[8,4,8,3]

$$\forall x\ (\ \exists y\ (\ F(\text{donkey})(\emptyset^{xy}_{12}(2));\ [\![\ \text{man}_{N^1}\ ]\!]^{\emptyset^{xy}_{12}} = 1\ \wedge\ [\![\ [_{PP}\ \text{with}_P\ \text{it}_{NP^2}]\ ]\!]^{\emptyset^{xy}_{12}}(\emptyset^{xy}_{12}(1)))$$
$$\exists z\ (\ F(\text{woman})(\emptyset^{xz}_{13}(3));\ [\![\ [_{VP}\ \text{loves}_V\ \text{it}_{NP^3}]\ ]\!]^{\emptyset^{xz}_{13}}(\emptyset^{xz}_{13}(1))))$$

[-,5,-,6]

$$\forall x\ (\ \exists y\ (\ F(\text{donkey})(y);\ [\![\ \text{man}_{N^1}\ ]\!]^{\emptyset^{xy}_{12}} = 1\ \wedge\ F(\text{with}_P)(\emptyset^{xy}_{12}(2))(\emptyset^{xy}_{12}(1)))$$
$$\exists z\ (\ F(\text{woman})(z);\ F(\text{loves}_V)(\emptyset^{xz}_{13}(3))(\emptyset^{xz}_{13}(1))))$$

[-,8,-,-]

$$\forall x\ (\ \exists y\ (\ F(\text{donkey})(y);\ F(\text{man})(\emptyset^{xy}_{12}(1))\ \wedge\ F(\text{with})(y)(x))$$
$$\exists z\ (\ F(\text{woman})(z);\ F(\text{loves})(z)(x)))$$

[-,-,-,-]

$$\forall x\ (\ \exists y\ (\ F(\text{donkey})y;\ F(\text{man})(x)\ \wedge\ F(\text{with})(y)(x))$$
$$\exists z\ (\ F(\text{woman})(z);\ F(\text{loves})(z)(x)))$$

3.7 Some Comparisons

3.7.1 The relation to Montague's account

Montague is often charged with analysing semantic ambiguity via an unmotivated syntactic ambiguity. In contrast, the syntactic account given above only assigns one syntactic structure to a sentence such as 'every man loves a woman'. In this section, I first outline Montague's official position and conclude that the charge against him is just; but discussion of a mild reformulation leads to a distinction between two different types of syntactic ambiguity for which Montague is guilty on only one count. Finally the reformulation is compared to our own account.

In [Montague 74b], a highly general method for dealing with ambiguous languages is defined. First, one defines a (free) syntactic algebra for a disambiguated language $\mathcal{U}$. This language is interpreted by a function g (homomorphism, in fact) into a semantic algebra. Secondly, one defines a *relation* R between expressions of $\mathcal{U}$ and expressions of the (ambiguous) language L that we wish to interpret. An interpretation for an expression in L, ζ , is given by its relation to an expression in $\mathcal{U}$, ζ', whose interpretation we already have. Taking Montague strictly at his own words, it can be shown that he does only allow a semantic ambiguity in virtue of a syntactic ambiguity. In the following definitions from [Montague 74b], $\mathcal{B}$ means the semantic algebra for $\mathcal{U}$ plus the meaning assignment to the basic expressions, and $\bigcup_{\delta\in\Delta} C_\delta$ is the set of all expressions generated by $\mathcal{U}$.

> **1)** ζ means b in L according to $\mathcal{B}$ if and only if there exists $\zeta' \in \bigcup_{\delta\in\Delta} C_\delta$ such that $\zeta' R \zeta$ and $g(\zeta') = b$
>
> **2)** ζ is semantically ambiguous in L according to $\mathcal{B}$ iff ζ means at least two different things in L according to $\mathcal{B}$
>
> **3)** ζ is syntactically ambiguous in L iff there are at least two objects $\zeta' \in \bigcup_{\delta\in\Delta} C_\delta$ such that $\zeta' R \zeta$

Now suppose that ζ is semantically ambiguous. Then, by **2** there are two objects a, b $(a \neq b)$ which ζ means. Therefore, by **1** and the fact that g is a function, there are two objects ζ_1', ζ_2' $(\zeta_1' \neq \zeta_2')$ such that $\zeta R \zeta_1'$ and $\zeta R \zeta_2'$. But if there are two objects that are related by R to ζ then ζ is by definition syntactically ambiguous.

In the fragment of English presented in [Montague 74b], the ambiguous language consists simply of strings of English and the members of the disambiguated language are quite naturally taken as syntactic structures. However, in [Montague 74a] a different approach is taken whereby 'analysis trees' are recursively defined whose nodes are annotated with strings of English. No disambiguating relation R is explicitly defined. Montague comments that we can consider the trees to be members of the disambiguated language and the annotations at the topmost nodes the members of the

ambiguous language. Once more, these annotations are strings and it is easy to see the analysis trees as syntactic structures. There is however the possibility of annotating the trees not with strings but with syntactic structures. The analysis trees will then record the derivational history of the syntactic structures but will not themselves be syntactic structures (*cf.* [Partee 76]). The position then would be that although 'every man loves a woman' is assigned *one* syntactic structure, that structure has alternative derivations. Cooper, in [Cooper 83] for example, defines a string to be 'strongly syntactically ambiguous with respect to a grammar' if it is assigned more than one syntactic structure and 'weakly syntactically ambiguous' if there is more than one derivation tree with the same syntactic structure at the topmost node. He presents a grammar with a rule of 'NP lowering'

NP lowering:
if $\alpha = [_{NP}\ \xi]$ and $\beta = [_{S}\ X\ [_{NP}\ x_i]\ Y]$ (X, Y may be empty)
then $[_{S}\ X\ [_{NP}\ \xi]\ Y']$ is a syntactic structure
(Y' is Y with all x_i replaced by he/she/it depending on ξ's gender)

Use of this rule permits at least two distinct types of derivation for sentences containing more than one noun phrase. Since the grammar *also* contains rules equivalent to $S \rightarrow NP\ VP$ and $VP \rightarrow V\ NP$, it follows that use of the NP lowering rule only produces syntactic structures which could equally well have been produced without it. The rule is syntactically redundant. Nevertheless, the string 'every man loves a woman' is only *weakly* syntactically ambiguous according to the grammar. It is not *strongly* syntactically ambiguous.

How does this compare with our account ? It would be a simple matter to claim that, so far as syntax goes, our account presents no syntactic ambiguity at all, either weak or strong, for this type of example. There is only one structure assigned to 'every woman loves a man' and only one way of assigning that structure. Although I think this is correct, it is also somewhat misleading because the way of assigning structure to a string is not also the way in which the sentence is semantically decomposed. Our system is not a 'rule-to-rule' system. The dialectical position then is this: we object

to Montague's official position on the grounds that he divines a syntactic ambiguity where there is none; we object to the reformulation using 'NP-lowering' since that system makes use of unnecessary syntactic rules; and our response is to remove those rules from the syntax and allow the semantics to be less tightly coupled to the syntax.

Is the result a defensible one? I think it is. By the lights of chapter 2, the system provides 'a finite characterisation of the possible syntactic forms of a language and ...how each of these forms determines the meaning which it in fact carries' (Kamp's demands). The theory supplies an effective explanation of the semantic role of each significant expression in any of its appearances (Davidson's demands). Can we not always answer the question 'What are these familiar words doing here ?'

There is of course the worry that we have not, for example, provided a *denotation* for the VP 'loves a woman'. We have only characterized how 'a woman' contributes to sentences in which it appears. This need not detain us, however, because the values assigned to *all* subsentential units are just logical constructions designed to give appropriate values, eventually, to the whole sentences in which they appear. There is no independent test of what value to assign to a subsentential unit other than what effect is produced on the sentences in which it appears. Another, more troublesome worry might be that the recursion does not actually proceed over *parts* of the sentence - since the substitution operation causes us to recurse over another sentence involving a different lexical item. Although this undoubtedly looks odd, it is simply an instance of the fact that the meanings of whole *sentences* are not independent of each other. The rules for assigning values to some sentences are sufficient in themselves to determine the values for certain other sentences - such is the nature of entailment. Why not allow this sort of fact to be written directly into the clauses governing the interpretation of quantified sentences? That is, 'every man runs' is true just in case for each man, 'it runs' is true in a context where 'it' denotes that man. The alternative, of course, is to state rules, perhaps *à la* Montague or *à la* Cooper (for which, see below), which have those facts only as *semantic consequences* of the definitions. But if the job in

hand is simply to predict the truth conditions of sentences then both approaches seem equally up to scratch. A preference for assigning some one value to each syntactic constituent will have to be based on something else besides 'compositionality'. (The issue of finding a non-syntactic reason for choosing between equally viable semantic theories - from the point of view of predicting truth conditions - is discussed further though by no means conclusively in [Evans 85] and [Lewis 81]. Cooper also suggests in [Cooper 83] that the rule-to-rule approach might be more acceptable in the search for *incremental interpretation*, which I discuss further in chapter 7.)

3.7.2 The Relation to Cooper Storage

The method of Cooper Storage is motivated directly by the need to avoid having unnecessary syntactic rules (such as 'NP-lowering') whilst also preserving the rule-to-rule hypothesis. Syntactic structures are assigned sets of interpretations and it is the set of interpretations for a structure which is determined compositionally by the sets of interpretations of its parts. An interpretation is an ordered pair containing a familiar meaning (for example, the set of all sets containing a woman if we are interpreting 'a woman') and a *Store*. A Store is a set of noun phrase denotations. The idea is to build up compositionally these two-part structures where noun phrase denotations have yet to be given scope and then have a special 'NP-retrieval' rule which will take a noun phrase denotation out of Store and give it scope over the first 'familiar' part of the interpretation. The order of removal from Store again determines relative scoping. NP denotations taken out earlier receive narrower scope than those removed later, just as in the 'basic' version of Hobbs and Shieber's algorithm.

A brief example will suffice to illustrate the procedure.Let $\langle\langle M, S\rangle\rangle$ be an ordered pair where S is the Store and M is the more familiar denotation. Suppose $\lambda P.\forall x\ (m(x) \rightarrow P(x))$ is the familiar denotation of 'every man'. I shall abbreviate this by $[\![\ every\ man\ ]\!]$. Then, 'NP-Storage' dictates that $\langle\langle [\![\ every\ man\ ]\!], \emptyset\rangle\rangle$ and $\langle\langle x_i, \{[\![\ every\ man\ ]\!]_i\}\rangle\rangle$ are members of the interpretation set for 'every man'. (The indices are used to link

the stored noun-phrase with the exact argument position it is to quantify over). In general, the rules which generate interpretations for phrase-structure construct the component M just as one would ordinarily an interpretation out of components. They also construct the component S for a structure by forming the set-union of the Stores of the parts of the structure (see [Partee & Bach 81] for interesting variations). So, if the interpretation of 'loves' is $\langle\langle loves', \emptyset\rangle\rangle$ then one interpretation of 'loves every man' is $\langle\langle loves'(x_i), \{[\![\ every\ man\ ]\!]_i\}\rangle\rangle$ (the other interpretation will be $\langle\langle \lambda P.\forall x\ (m(x) \rightarrow loves'(x)), \emptyset\rangle\rangle$). Furthermore, one interpretation of 'a woman loves every man' will be $\langle\langle loves'(x_i, x_j), \{[\![\ every\ man\ ]\!]_i, [\![\ a\ woman\ ]\!]_j\}\rangle\rangle$. At this point 'NP-retrieval' can be invoked which generates more interpretations for the sentence. The rule is :

NP-retrieval
if $\langle\langle M, S\rangle\rangle$ is an interpretation for a sentence and $\sigma_k \in S$,
then $\langle\langle \sigma(\lambda x_k.M), S - \{\sigma_k\}\rangle\rangle$ is also an interpretation for the sentence.

We can therefore derive the following two interpretations for our example:

$\langle\langle [\![\ every\ man\ ]\!](\lambda x_i.[\![\ a\ woman\ ]\!](\lambda x_j.loves'(x_i, x_j))), \emptyset\rangle\rangle$
$\langle\langle [\![\ a\ woman\ ]\!](\lambda x_j.[\![\ every\ man\ ]\!](\lambda x_i.loves'(x_i, x_j))), \emptyset\rangle\rangle$

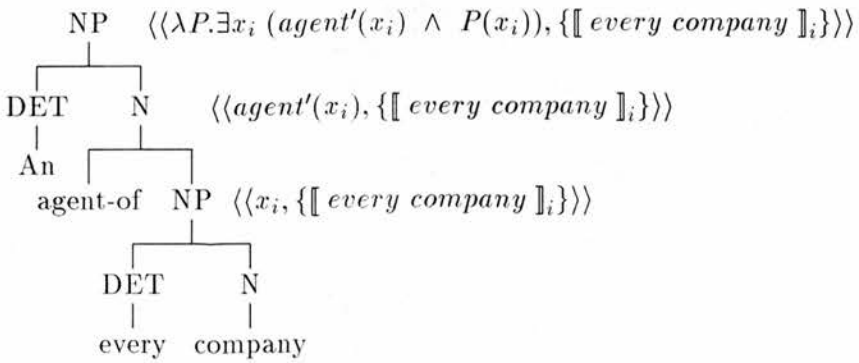
(In fact, the use of indices, particularly in a presentation such as this one, has generated doubts over whether the method is compositional. For example, when we apply a member of store to the first part of a denotation, we must look to a purely syntactical feature of the expression denoting the member of store, namely what index it has (the index has no interpretation itself). For this particular debate, I refer the reader to [Landman 83] for some criticism, to [Cooper 83] which uses a different method of indexing, and to [van Eijck 85] for some discussion).

It will be recalled that one of Hobbs and Shieber's claims for their algorithm was that it could handle complex noun phrases better than previous accounts, including Cooper's. Cooper Storage overgenerates for complex noun phrases because the NP-retrieval rule is designed to give the stored NP-denotation scope whilst also ensuring that that it

correctly binds the argument position from whence it came. This is why one applies the stored denotation to a lambda-abstraction over the variable with which the store was indexed. However, if a sentence contains a complex noun phrase then an index to be bound can itself end up inside the store and may not be present in M when the associated indexed NP-denotation is applied.

Below is a representation of the NP 'an agent of every company' where interpretations of the two NP nodes and the N node are given to the right of the node name. In the example, 'every company' has been stored but 'an agent of every company' has not.

(21)



If we now apply the NP-Storage rule to the topmost node the result is 22.

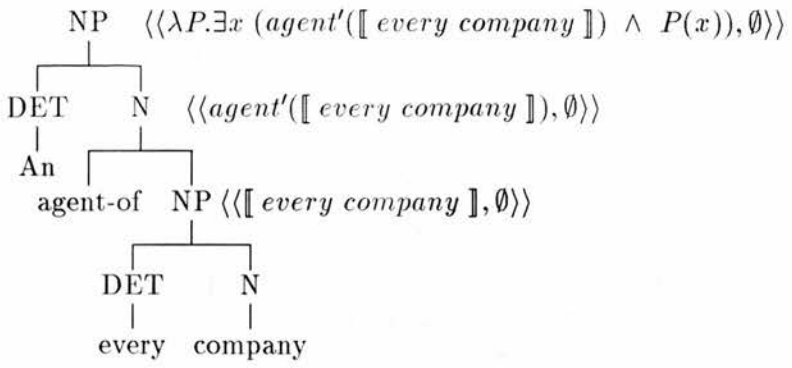
(22) $\langle\langle x_j, \{\lambda P.\exists x_i\ (agent'(x_i)\ \wedge\ P(x_i))_j, [\![\ every\ company\ ]\!]_i\}\rangle\rangle$

Suppose this store is simply propagated up the tree, then the first part (M) of the interpretation for the S-node will not contain x_i. Application of NP-retrieval using the NP denotation indexed with i now results in a problem because x_i is not present in M and will not be bound. Subsequent application of the other NP-denotation will then lead to the presence of a free variable. This is of course precisely analogous to the error we found in the 'basic' version of Hobbs and Shieber's algorithm. Their response was to forbid the application of nested complex terms and also to abandon the 'inside-out' policy (NPs chosen earlier receive relatively narrower scope) at this point only. We have recommended that simply adopting an outside-in policy throughout is sufficient to deal with the problem.

[Keller 86] explores a similar solution, whilst keeping to the general framework of 'Cooper Storage'. Keller adds structure to the stores so that the denotations of embedded NPs never get taken out of store before those of the embedding NPs. The idea is that when one stores an NP, then one places in store the *whole interpretation* $\langle\langle M, S\rangle\rangle$ and not just M. Stores may now contain objects which themselves contain stores - so the idea is named 'Nested Cooper Storage'. The structure of the stores now reflects structure in the NPs. It remains the case that one applies members of the store to M using NP-retrieval - but embedded NPs will not have their denotations as *members* of store (they will be nested within other members' denotations) so outermost NPs are always applied first. There is an amendment to NP-retrieval so that the new store is no longer $S - \{\sigma\}$ but $(S - \{\langle\langle M_\sigma, S_\sigma\rangle\rangle\}) \cup S_\sigma$. The nested members of σ are thereby promoted up one level of nesting.

The result of Nested Cooper Storage is similar to that of Hobbs and Shieber because the nesting in the stores means that embedded stores (complex terms) cannot be applied before the store (complex term) which embeds them. With nested stores (complex terms) there is only one possible order of application of NP-denotations. However, in Hobbs and Shieber's case, this led to a problem because we still needed to generate an ambiguity. Therefore, there was the additional step of allowing an 'outside-in' recursion for the restrictions of complex terms. This is quite different from Keller's position. Keller generates the alternative reading by *not storing* the embedded noun phrase at all. That is, the general technique of Storage has *two* sources of indeterminacy. One is the choice of whether to store a particular NP or not; the other is in what order to unpack the store. If we do not store the embedded NP, then our example becomes

(23)



In contrast, Hobbs and Shieber want to generate all the disambiguations starting from only one representation. If one is tempted to think of Hobbs and Shieber's algorithm as encoding the unpacking of a Cooper-Store, then it is worth remembering that in order to generate all the meanings for a sentence, it is sometimes necessary to unpack *different* Cooper-Stores for the very same sentence structure.

Our own account, of course, proceeded from exactly one input representation and it did not need to invoke anything like the representational complexity of 'Nested Cooper Storage'.

3.7.3 Categorial Accounts

Accounts of quantifier scoping have also recently been given in the framework of categorial grammar ([Hendriks 90],[Emms 90]). I shall not here describe their proposals in any great detail but restrict myself to two brief remarks.

First, as I indicated earlier, the system of [Hendriks 90] uses sets of meanings in place of a single meaning. So when a structure is assigned a meaning (either lexically or as a result of application of a construction rule), then it is also assigned many, possibly infinitely many, other meanings as a result of possibly repeated applications of three type-shifting rules. These rules are of the form that 'if you have a meaning whose type pattern-matches α, then you also have a related meaning β', where β may contain abstractions over variables whose type depends on the original pattern-matching. Not all the extra meanings are propagated through a derivation however as the rules of

combination may only apply to certain members of the sets of meanings of the constituent parts. The generation of 'extra' meanings for the same structure by means of rules of form 'if X is a meaning for P, so is Y' resembles Cooper's analysis.

A rather different approach is taken by Emms. In categorial grammar the set of categories is inductively defined over a base set which usually contains just s and np. Complex categories are formed by a rule: 'If a and b are categories then so is $a \oplus b$, where $\oplus$ may be one of $/, \backslash, |$ (right-seeking, left-seeking and non-directional). s/np is the category which will combine with an np to the right to produce an s. Emms extends the categories to include polymorphic categories. For example there is a category $\forall X.X/(X \backslash np)$, which is a category that could combine with a category $s \backslash np$ to the right to produce an s or with a $vp \backslash np$ to the right to produce a vp. The semantics is 'strictly' compositional by assigning precisely one meaning to each category that appears in a derivation and assigning one semantic combination rule for each axiom governing syntactic derivation. It is a little harder to compare the resulting system with the others since Emms does not make use of a similar notion of syntactic structure. For example, Hendriks is concerned to combine a transitive verb with its object first to produce a structure which can then combine with its subject. Emms has no such ambition. His purpose is to show that given suitable lexical entries, his categorial calculus predicts that a multiply quantified sentence is classified correctly as an s and that it receives all and only the correct meanings. The different meanings arise from different proofs that the string is a sentence. The structure of the proofs and the order of combination within the proofs is not given any special syntactic significance. Whether one combines a transitive verb with its subject or object first is not a significant question - in fact, one can derive quantifier scope ambiguities with either combination.

One point we might note however is that Emms is concerned to have an 'emergent' account of quantifier scoping - by which he means that there should be no special additional rules in the system which are designed to account for quantifier scope am-

biguities. The rules which govern simpler constructions should allow ambiguities in multiply quantified sentences to emerge from that very multiplicity. A similar claim might be made for the account I have proposed - for the indeterminism present in the rules is only significant when there is a multiplicity of possible rule applications. The very same device is used for the simpler and the more complex cases - and the device is not itself a more complex one than one would otherwise employ.

3.8 'Higher Order' Constructions

3.8.1 Opacity and Conjunction

Hobbs and Shieber's algorithm is designed not only for relative quantifier scopings but also for quantifier interactions with 'opaque arguments of higher order predicates'. I have already commented that it is unclear what notion of 'opacity' is being used here. The use to which it is put can however be understood as a response to another problem generated by the 'inside-out' strategy embodied in the basic step of Hobbs and Shieber's algorithm. Suppose we represent 24a by 24b.

(24) a Everyone isn't here
b not(here($< \forall\ x$ person(x)$>$))

The basic algorithm will apply the embedded complex term to the whole formula thus giving 'every' scope over the whole formula including the negation. The question is then how to give 'every' narrow scope with respect to the negation, since one reading of 24a is that it is not the case that everyone is here. Hobbs and Shieber respond by devising a new predicate, 'opaqueness' which is a predicate of argument positions. Negation is said to be opaque in its only argument. When an input structure has opaque arguments then those arguments are both fully and partially scoped, and the results substituted for the original arguments before the basic algorithm is applied to the resulting entire expression. A partial scoping is simply a result where not all complex terms have been removed from the input. As an example of the

handling of opaque arguments consider 24b. A full scoping of the opaque argument results in '(every,x,person(x),here(x))', whereas a partial scoping results in 'here(< every x person(x)>)'. These two results are then substituted for the original arguments to the predicate generating 'not(every,x,person(x),here(x))' and 'not(here(< every x person(x)>))'. The basic algorithm is now executed with these two inputs. The first representation contains no complex terms now so no modifications are made by the basic algorithm. The overall result is therefore one where 'not' outscopes 'every'. When the second representation is passed to the basic algorithm, the complex term will be extracted 'widely' resulting in the reading where 'every' outscopes 'not'. Therefore, the two readings of 24a have both been generated.

It is a moot point whether Hobbs and Shieber's mechanism actually delivers the right results in all cases. For example, 'every man doesn't love a woman' is predicted to have a reading where the negation takes widest scope, followed by the existential quantifier and then the universal. This reading constitutes the *denial* that there is a woman whom every man loves. This is a not a reading which Montague predicts in [Montague 74a]. The reason is that Montague only allows one to introduce a negation when combining a term (or noun phrase) with an intransitive verb phrase. The only way to give 'a woman' wide scope over 'every man' is to quantify it into the sentence 'every man loves her'. But the result of this is another *sentence* so there is no longer an appropriate intransitive verb phrase to negate. (Quantifying 'a woman' into 'every man doesn't love her' results in 'a woman' having wide scope over the negation). Here, I shall assume that negation does operate simply as an operator on sentences.

The figure below is another illustration of Hobbs and Shieber's two-stage processing for opaque predicates. The figure illustrates the processing of 25a as represented by 24b. (For clarity, I omit the restrictions of the complex terms).

(25) a jack believes everyone loves someone
b bel(j,loves(< $\forall$ x >,< $\exists$ y >))

The two-place predicate 'believes' is marked as opaque in its second argument. Therefore the second argument 'loves($< \forall\ x >, < \exists\ y >$)' will be fully and partially scoped before the basic algorithm is set to work. There are five possible ways of (partially) scoping the complex terms in this expression (scope neither, scope one, scope the other, and two ways of scoping both). The results are shown in the second column of figure 1. The results of the basic algorithm operating on these five representations are shown in the third column.

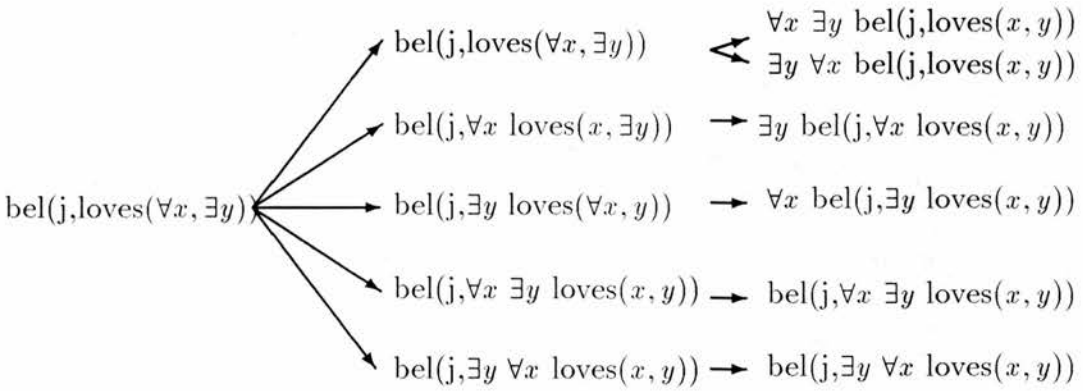


Figure: 1

The two-stage procedure can be objected to on two grounds. First, it is again difficult to interpret the workings of the algorithm since the outputs of the first stage are formulae which are neither wholly in the input (object) language nor in the output (meta) language. Secondly, the overall policy is that, by default, quantifiers are given as wide a scope as possible and that we *then* worry about how to give them a narrower scope. When we come to consider the possible limits on wide-scoping however, we will have to take two actions. One will be to disable the 'widest-scope' reading and the other will be to enable a 'narrower-scope' reading. For example, if we consider sentence conjunction, we find that we do not want to extract quantifiers across such a conjunction. In fact, this thesis is partly concerned with using *dynamic mechanisms* to secure sentence-external quantifier dependencies, rather than some wide-scope rule. Even disregarding this however, we still do not want to use a wide scope rule. For example, 26a should not be analysed via 26b.

(26) a Every student came to the party and Sue was pleased
b Every student is such that (they came to the party and Sue was pleased)

The reason is that Sue's displeasure is sufficient to refute 26a but not 26b - should there in fact be no students. In order to prevent wide-scoping and allow narrow scoping of 'every student' in Hobbs and Shieber's algorithm, we would have to deem both that a complex term embedded in a conjunctive formula is not applicable to that formula and mark 'and' as being opaque (in both its arguments).

By way of contrast, let us add a simple rule for conjunction to the semantic theory for HSL (section 3.4.1).

5 $[\![and(\phi, \psi)]\!]^g = 1$ if $[\![\phi]\!]^g = 1 \wedge [\![\psi]\!]^g = 1$

Furthermore, we stipulate that $\Sigma(< \exists\ x\ \phi >_i)$ no longer denotes *any* HSL formula containing $< \exists\ x\ \phi >_i$ but only those where $< \exists\ x\ \phi >_i$ is not itself contained within a formula of form $and(\pi, \psi)$. That is, a complex term may not be 'extracted' if it occurs within a conjunction. This is sufficient to produce the same results as Hobbs and Shieber's algorithm. The difference is that having disabled the wide scope reading, no additional machinery is needed to generate the narrow scope readings.

3.8.2 Negation

Substantially the same policy will work for negation too. However, there is a complication here owing to our use solely of conditionals rather than biconditionals in the truth theory. The natural clause to adopt for negation is the following:

6 $[\![not(\phi)]\!]^g = 1$ if not($[\![\phi]\!]^g = 1$)

and one might also naturally write down the following algorithmic rule

Negation
if the input is of form 'not(ϕ)',
recursively scope ϕ producing ϕ'
return 'not(ϕ')'

Somewhat surprisingly, this is not correct. The difficulty is caused by the *valid substitutions* that one may make in the meta-language. For example, we know

A: $[\![\text{loves}(< \text{every } x \text{ man}(x) >, < \text{some } y \text{ woman}(y) >)]\!]^g = 1$
if every(x,man(x),some(y,woman(y),loves(x, y)))

but we cannot infer by rule 6 that

B: $[\![\text{not}(\text{loves}(< \text{every } x \text{ man}(x) >, < \text{some } y \text{ woman}(y) >)]\!]^g = 1$
if not (every(x,man(x),some(y,woman(y),loves(x, y))))

The reason is simply that since 'loves(<every x man(x)>,<some y woman(y)>)' is true in *either* of two conditions the fact that *one* of those conditions does not hold is insufficient to draw the required conclusion. **A** and rule 6 is not enough to conclude **B**. So the 'natural' algorithmic interpretation of rule 6 does not represent a valid inference in the truth theory.

In order to justify a substitution within the scope of a negation in the meta-language, we appear to need, after all a version of rule 6 which uses 'only if' as well as 'if'. Of course, we can obtain one by the means we suggested earlier: once we know that 'p is true if π', that 'p is true if ψ' and that there are no *other* ways in which p can be true then we can conclude 'p is true iff (π or ψ). That is, the truth theory for HSL plus rule 6 allows us to derive the following

(27) not($\exists\ \alpha$(woman(α);$\forall\ \beta$(man(β);loves(α, β))) $\vee$
$\forall\ \beta$(man(β);$\exists\ \alpha$(woman(α);loves(α, β))))

Interestingly, this is not a reading which Hobbs and Shieber's algorithm produces. After all, Hobbs and Shieber's algorithm, just like the amended version I have proposed, never introduces a connective into an output representation which is not present in the input representation, yet the truth condition in 27a contains a disjunctive connective.

My own opinion is that 27 is *not* a reading of 'every man doesn't love a woman', even when negation is a sentential operator. The reason is that if what a speaker intends by

uttering π is A, then surely what he intends by $not(\pi)$ is $not(A)$. Even if π is actually ambiguous between A and B, a hearer who spots the ambiguity will reason that a speaker of π intended one of the pair $< A, B >$, and therefore that what a speaker intended by $not(\pi)$ was one of the pair $< not(A), not(B) >$. He will not reason that the speaker intended $not(A \ or \ B)$, just as he will not think that a speaker intends $A \ or \ B$ by uttering π. There is perhaps one possible exception to this - where the language users are semantic theorists who are gripped by the correctness of rule 6 for negation or the idea that π must *mean* $A \ or \ B$. The analysis of the language of semantic theorists in the grip of a false theory of English must await another occasion.

The conclusion is that rule 6 is not the right rule for negation when we are using a truth theory which uses only conditionals and not biconditionals.

3.8.3 Processing Negations using Conditionals only

Instead of using rule 6, we will adopt the method for handling negation used in partial logics, *i.e.* by distinguishing the conditions for truth from those for falsity. The idea is that 'not(ϕ)' is true if 'ϕ' is false. Also, 'not(ϕ)' is false if 'ϕ' is true. In this way, the meta-language in which we state the theory only *uses* negation at the very base of a recursion. Therefore, we never have to worry about which substitutions are valid within the scope of a meta-language negation.

The semantic effect of an object language negation will not be to introduce a meta-language negation, as was the case before. Rather, it will cause us to flip between looking for truth conditions and looking for falsity conditions. For each syntactic construction we now provide, additionally, a meta-language statement of the conditions in which a formula is false. For example, 'p(t_1 .,.,., t_n)' is false w.r.t. g if $< g(t_1).,.,.,g(t_n) > \notin F(p)$

Here is the final truth theory for $HSL+$ (HSL plus conjunction and negation).

HSL+syntax

The following six clauses define the pre-formulae of *HSL*+.

1. There is a basis set containing individual constants, n-ary predicates and an infinite number of variables.
2. If $t_1, \ldots, t_n$ are individual constants, variables or complex terms and P is an n-ary predicate, then $P(t_1, \ldots, t_n)$ is a formula.
3. If ϕ is a formula, and x is a variable, then $< \exists\, x\ \phi >$,$< \forall\, x\ \phi >$ and $< Most\ x\ \phi >$ are complex terms.
4. If ϕ is a formula, then so is 'not(ϕ)'.
5. If ϕ and ψ are formulae, then so is 'and(ϕ, ψ)'.
6. Nothing is a formula or complex term except in virtue of the above clauses.

A formula of *HSL*+ is any pre-formula where no two complex terms bind the same variable. If Σ is a formula containing a particular complex term π, and π occurs in no formula of form 'and(Φ,Ψ)' within Σ, then we may also denote Σ by $\Sigma[\pi]$.

HSL+ semantics

A model for *HSL*+ is a pair $< D, F >$ where D is a non-empty set of individuals and F is a function from individual constants to members of D, and from n-ary predicates to n-ary relations on D. Assignments are partial functions from variables to members of D.

$[\![\ x\ ]\!]^g$ is $g(x)$ for all variables x.

1a $[\![P(t_1,\ldots,t_n)]\!]^g$ $=1$ if $< [\![t_1]\!]^g,\ldots,[\![t_n]\!]^g > \in F(P)$
b $=0$ if $< [\![t_1]\!]^g,\ldots,[\![t_n]\!]^g > \notin F(P)$
(where t_i are variables or individual constants)

2a $[\![\Sigma\lceil < \exists\ x\ \phi > \rceil]\!]^g$ $=1$ if $\exists\alpha\ ([\![\phi]\!]^g = 1; [\![\Sigma[x/ < \exists\ x\ \phi >]]\!]^g = 1)$
b $=0$ if $\forall\alpha\ ([\![\phi]\!]^g = 1; [\![\Sigma[x/ < \exists\ x\ \phi >]]\!]^g = 0)$

3a $[\![\Sigma\lceil < \forall\ x\ \phi > \rceil]\!]^g$ $=1$ if $\forall\alpha\ ([\![\phi]\!]^g = 1; [\![\Sigma[x/ < \forall\ x\ \phi >]]\!]^g = 1)$
b $=0$ if $\exists\alpha\ ([\![\phi]\!]^g = 1; [\![\Sigma[x/ < \forall\ x\ \phi >]]\!]^g = 0)$

4a $[\![\Sigma\lceil < \mathit{Most}\ x\ \phi > \rceil]\!]^g$ $=1$ if Most α $([\![\phi]\!]^g = 1; [\![\Sigma[x/ < \mathit{Most}\ x\ \phi >]]\!]^g = 1)$
b $=0$ if Nmst α $([\![\phi]\!]^g = 1; [\![\Sigma[x/ < \mathit{Most}\ x\ \phi >]]\!]^g = 0)$

5a $[\![\mathit{and}(\phi,\psi)]\!]^g$ $=1$ if $[\![\phi]\!]^g = 1\ \wedge\ [\![\psi]\!]^g = 1)$
b $=0$ if $[\![\phi]\!]^g = 0\ \vee\ [\![\psi]\!]^g = 0)$

6a $[\![\mathit{not}(\phi)]\!]^g$ $=1$ if $[\![\phi]\!]^g = 0$
b $=0$ if $[\![\phi]\!]^g = 1$

The only unclarity in the definition resides in the meaning of the meta-language quantifier 'Nmost'. Naturally it is intended to represent the dual of 'most' but as pointed out by [Barwise & Cooper 82], there appears to be no simple natural language dual for 'most'. A reasonable approximation is 'At least half'.

If one now recurses through the clauses for 'every man doesn't love a woman' represented by 28a, then one will find the same six readings that Hobbs and Shieber predict, including the readings discussed above 28b and 28c.

(28) a not(loves(<some x woman(x)>,<every y man(y)>))
b $\exists\ \alpha$(man(α);$\forall\ \beta$(woman(β);not(loves(α,β))))
c $\forall\ \beta$(woman(β);$\exists\ \alpha$(man(α);not(loves(α,β))))

The resulting theory is therefore one that agrees with Hobbs and Shieber's predictions for negated and doubly quantified sentences. They are the predictions which appeared to be most natural and yet not to be generated by the 'obvious' semantic rule for interpreting negation.

3.8.4 An *HSL+* Algorithm

The algorithm for generating readings based on *HSL+* differs from our previous algorithm in the following respects. First, *apply-terms* now takes two arguments where the second argument *polarity* always evaluates either to **1** or **0**. This corresponds to the search for truth or falsity conditions. Secondly, *term* returns true only if there is a complex term in *form* which is not embedded within a conjunction; and *applicable-term* only returns complex terms that are not embedded within conjunctions. The function *dual* is a 2-ary function that takes an HSL quantifier or connective and a polarity and returns that same expression if *polarity* is **1**; otherwise it returns the expression's dual, *i.e.* $\forall$ for $\exists$, *or* for *and* and so on. *conj* is a function that takes an HSL formula as argument and returns *true* if the formula is conjunctive (otherwise *false*). *neg* is a function that returns *true* if its argument is a negated HSL formula (otherwise *false*. *rev* is a function that returns **1** if its argument is **0** and returns **0** if its argument is **1**. *cwff* that takes a connective and two formulae as arguments and returns one formula containing the connective and the two arguments

```
function apply-terms(form,polarity);
  if      term(form)
  then    let < quant var restrict> := applicable-term(form)
          in wff(dual(quant,polarity),
                 var,
                 apply-terms(restrict,1),
                 apply-terms(subst(var,< quant var restrict>,form),polarity))
  else
          if      conj(form)
          then    let and(first,second) := form
                  in cwff(dual(and,polarity),
                          apply-terms(first,polarity),
                          apply-terms(second,polarity))
          else
                  if      neg(form)
                  then    let not(exp) := form
                          in apply-terms(exp,rev(polarity))
                  else    if polarity = 1
                          then form
                          else wff(not(form))
```

(A program implementing this algorithm can be found in Appendix A.2).

3.9 Conclusion

In this chapter, we have presented a method for defining truth theories for languages which contain quantifier scope ambiguities. The theory consists of a recursive definition of satisfaction which uses only conditionals and not biconditionals. The theory makes no appeal to syntactic ambiguities, nor to the complexities of Cooper Storage. The cost of this is that the resulting system is not a rule-to-rule system, but it is compositional by the lights of chapter 2. Furthermore, the theory naturally suggests a quantifier scoping algorithm which bears a very close relation, though not identity to that of Hobbs and Shieber's algorithm. Working ones way through the algorithm can be thought of straightforwardly as working ones way through the associated truth theory. In this way, the success of the algorithm is explained. The algorithm was also extended to cover extensional 'higher order' predicates including conjunction and negation. In order to continue using only conditionals when negation is added to the object language, we adopted a standard practice from partial logics in distinguishing truth conditions from falsity conditions. An algorithm for the resulting theory was also presented.

Chapter 4

Pronoun Resolution in Dynamic Semantics

4.1 Introduction

It is an outstanding question in Dynamic Logic how one is to determine which antecedents a pronoun may have. If we consider 29

(29) A (certain) student had been seeking his roots for a long time. Then, the shopkeeper described a man who had known his mother

and concentrate just on the second sentence, devoid of the discourse context in which it is set, then one would traditionally identify at least the following two truth conditions 30a,b.

(30) a The shopkeeper described a man who had known the shopkeeper's mother
b A man who had known his own mother was described by the shopkeeper

When we allow sentence-external influences (broadly, 'contextual' influences) to operate too then there is also the possibility of deixis whereby 'his' is accompanied by a pointing at some perceptually salient object. The second sentence is then true if the shopkeeper described a man who had known the mother of the object pointed at. There is, furthermore, the possibility of a 'linguistic context' created by the interpretation of what has already been said which can supply us with further possibilities for the reso-

lution of pronouns. It is these, dynamic, cases that we are particularly concerned with here. Our example 29 is most naturally interpreted using such a possibility, generating the following truth condition

(31) There is some student who had sought his roots for a long time and whose mother was known by some man who the shopkeeper described.

These sorts of ambiguities have not been predicted for the formal languages described so far in this thesis, for Groenendijk and Stokhof's DPL or for Hobbs and Shieber's input language. The reason is that they all assume that the decision concerning which noun phrase a pronoun occurrence has for its antecedent has already been taken. For example, one of the key motivations for Groenendijk and Stokhof's dynamic semantics is precisely to predict that 32a *as formalized by* 32b has a semantics as paraphrased by 32c.

(32) a Every farmer who owns a donkey feeds it
b $\forall x\ ((f(x) \wedge \exists y\ (d(y) \wedge o(x,y))) \rightarrow f(x,y))$
c Every farmer who owns a donkey feeds any donkey he owns

The formalization 32b already encodes the anaphoric dependency of 'it' on 'a donkey'.

Similarly, Hobbs and Shieber state that 33a has the structure 33b (for which I provide a suitable instantiation in 33c.

(33) a Every man with a picture of himself has arrived.
b $p(< q_1\ x\ r_1(x, < q_2\ y\ r_2(x,y) >) >)$
c arrived(<every x and(man(x),with(x,<a y picture(x,y)>)>)

Once again, 33b already encodes the decision that the quantified noun phrase is the antecedent of 'himself'.

Neither Groenendijk and Stokhof nor Hobbs and Shieber discuss how these decisions are arrived at. I should perhaps stress that the issue is not the lack of a means for identifying the *most plausible* of a list of candidate noun phrases. It is the lack of a means for identifying the class of possible candidates which is being objected to. That

is, do we have a reason for thinking that 'a donkey' is a possible antecedent for 'it' in 32a ? This is a question to which Discourse Representation Theory does, for example, supply an answer. DRT claims that 'a donkey' is a possible antecedent for 'it' in 32a because, at the point of processing 'it' there is a discourse referent introduced by 'a donkey' which is in an *accessible position* from the DRS in which we process 'it'. The question is not answered, explicitly at any rate, in [Groenendijk & Stokhof 91b] though we can perhaps cull an answer from what is made explicit there. If the question were to remain unanswered then doubt would be cast on the claim that Dynamic Logic is a compositional *alternative* to DRT. How could it be an alternative if one question which DRT answers (and which it deems important to answer) is left untouched in Dynamic Logic?

There is a further important issue that must be addressed here and that concerns the role of syntactic constraints in pronoun resolution. For example, in syntactic theories in the transformational tradition, syntactic rules are sought which predict when pronouns are bound by antecedents. In these theories, 'binding' generally means co-indexation of noun phrases and the possibility of binding is explained by the structural relations holding between noun phrases in syntax. In DRT, as we saw in chapter 2, the relation between a pronoun and its antecedent is not based primarily on syntactic relations between a pronoun and its antecedent but on relations between discourse referents and DRS conditions within DRSs. Syntactic constraints will still play a role in DRT (for example, in order to explain the complementary distribution of reflexive and non-reflexive pronouns) but this role will not exhaust the explanation of anaphoric possibilities.

The rest of this chapter is structured as follows. First, I discuss two ways in which Dynamic Logic might determine what the possible antecedents for a pronoun are but find both of them unsatisfactory. Neither makes direct use of the idea of evaluation in a context. Secondly, as a preliminary to my own positive proposal, I discuss what the proper notion of context should be in a dynamic theory. The positive proposal, which

does make direct appeal to the dynamic notions of evaluation in a context and context generation, is then specified and some illustrative worked examples are given. The final example illustrates the interaction between the treatment of quantifier scoping of the last chapter and that of pronoun resolution in this one. In the next chapter I will describe an algorithm based on the proposal.

4.2 Indexing the Syntax

4.2.1 Free Indexing

One possible way to specify which antecedents a pronoun may have is to employ *free indexing*, subject to certain conditions. Syntax may enjoin certain possible indexings (for example, Condition A of Binding Theory states that an anaphor - including, in particular, reflexive pronouns - must be bound in its minimal governing category *cf.* [Chomsky 81]), and also forbid others (for example, Condition B states that a pronominal - including non-reflexive pronouns - cannot be bound in its minimal governing category) but, aside from these cases, anything goes. So, in 32a, 'it' may have 'a donkey' as an antecedent just because it is not *forbidden.* The difficulty with this proposal is simply that it is not true that anything goes. Part of the range of data which DRT and DPL is held to account for is the following contrast.

(34) a *A man* came in. *He* sat down.
b * *Every man* came in. *He* sat down.

DRT has a story to tell about these cases. In 34a but not in 34b, when we begin processing the second sentence, there is an accessible discourse referent (introduced by 'a man') with which the discourse-referent for 'he' can corefer. In a free-indexing regime, why should we not simply co-index 'every man' and 'he' in 34b? Of course, it may be that we should not co-index them but we simply have not found the rule to prevent the co-indexing. Admitting this is just admitting that we cannot (yet) answer the question why 'he' may take 'a man' as antecedent in 34a and yet not take 'every man' as antecedent in 34b.

Alternatively, one might argue: "Why *not* co-index 'every man' and 'he' in 34b ? If you do, you will find that the result is, according to DPL, semantically uninterpretable. By contrast, co-indexing 'a man' and 'he' in 34a produces a result that *is* semantically interpretable. That is the difference between them." The difficulty with this reply is simply how much of an *explanation* it amounts to. Are we to suppose that syntax delivers a range of possible indexings and then, unfortunately, some just *turn out* to be uninterpretable ? In particular, human interpreters face the task of deciding on a suitable interpretation for a pronoun but where does this decision fit into the story according to dynamic logic? In DRT, when we face this decision there may be a number of possible options for resolution but none of these will turn out to be uninterpretable. The options for interpretation are all genuine possibilities based on the context (DRS) generated so far but this important aspect of dynamic semantics seems to be lost in an account which just employs free indexing.

4.2.2 Using Active Quantifiers in a Discourse

The second possible answer to our question is to characterize syntactically what the options for pronoun resolution actually are in dynamic logic. For example, as Kamp points out in [Kamp 90], there is a DPL analogue to the DRT notion of 'accessible discourse referent' in the DPL-notion of 'active quantifier' (see section 2.2 above). In 34a but not in 34b, there is an active quantifier for the whole discourse. The definition of 'active quantifier' is purely syntactic, so we could have a rule along the lines of 'co-index a pronoun with an active quantifier in the previous discourse'.

There are two reasons why such an approach looks implausible. One reason is simply that the definition of the class of active quantifiers is itself (syntactically) highly unnatural. In fact, it is quite evident that the definition is far more a record of what happens in the semantics than being an independently interesting syntactic notion. For example, we cannot appeal to configurational properties of trees in applying the notion. The special notion of binding in DPL depends crucially on facts about which

particular quantifier and which particular connective is used in a syntactic structure (though, to be sure, lexical properties of noun phrases enter also the Binding Theory Conditions of Government and Binding Theory - Condition C deals with noun phrases that are 'full-referring expressions'). Perhaps one could try re-coding the syntax so that it reflected the underlying dynamic semantic categories more closely but the result will be simply to point out how unnatural the resulting syntax is. 'All' and 'some' will not belong to the same category and neither will 'and' and 'or'. As Strawson commented on another topic

> One might surely have expected that if any pair of non-synonymous expressions exhibit non-difference of type or category, "or" and "and" would be one such pair and "all" and "some" another.([Strawson 70] pg. 184)

Perhaps this is the price that must be paid for dynamic semantics.

The second reason why the approach is implausible becomes apparent if we allow that sentences contain quantifier scope ambiguities and that those sentences are not also syntactically disambiguated. In such a case we will not be able to tell what the active quantifiers are until we have already embarked on a particular interpretation route. Barwise calls his example 7 (repeated below as 35)

(35) Statistics show that every 11 seconds a man is mugged here in N.Y. city. We are here to interview him

a 'jungle path sentence' precisely because we naturally assign a reading to the first sentence where 'every' takes wide scope and then we find we cannot interpret the pronoun 'him' on this reading. A jungle path sentence is a purely semantic analogue of the more familiar garden path sentence where a hearer naturally chooses one way to resolve a local ambiguity and cannot later revise his choice without a certain amount of difficulty.

4.2.3 Evaluating Pronouns in a Context

I want now to explore a third way to determine what antecedents a pronoun may have - one which places the notion of interpretation *in* a context rather more centre stage than it has appeared in the two earlier proposals. In chapter 1, it was stressed that it was precisely the notion of evaluating an expression in a context and producing a new context for subsequent evaluations which was distinctive in dynamic semantics. It is, of course, hardly an original idea that cases of discourse anaphora should be treated by appeal to a discourse context. The difficulty has always been gaining a good enough notion of discourse context to handle the sorts of case preeminently handled by DRT, such as 36a,b

(36) a A man came in. He sat down.
b Every farmer who owns a donkey feeds it

In the next two sections, I discuss first the idea of discourse contexts as quasi-perceptual contexts and secondly what concept of discourse context there is in DPL.

Discourse contexts as quasi-perceptual contexts

The notion of discourse context (outside of DRT) has often, I think, been assimilated to that of perceptual context, from which pronouns may also take values. Just as an utterance of 'he' may be accompanied by an act of pointing and thence take any object in the perceptual context as value, so, it may be thought, a discourse occurrence of a pronoun may take as value any object that has been introduced into the discourse context. Objects are introduced into the discourse context by being *referred to.* Lasnik in [Lasnik 76] suggests that the coreferential reading of 37

(37) John thought that he would win

is not secured by any syntactic rule but simply by the fact that since John has been referred to (by the initial use of 'John'), so 'he' may also refer to John. The hearer

or reader has to search the discourse context for suitable candidate referents, just as a stressed use of 'he' as in 38

(38) Look, *he* is wearing one of those new hats

will invite the hearer to search the perceptual context for a suitable referent.

A similar idea occurs in [Reinhart 83] where discourse anaphoric dependencies are again taken to be *referential.* Reinhart's idea is that there are only two sorts of pronoun - bound pronouns and discourse pronouns. The former cases are regulated by syntactic rules but the latter are free, barring certain pragmatic influences, to refer to salient objects.

(39) a Felix thinks that he is a genius
b He thinks that Felix is a genius

In Reinhart's system, coreference between 'Felix' and 'he' in 39a is secured by a rule. In 39b, 'He' is free to refer to any salient object, including Felix. Reinhart invokes a pragmatic rule to prevent the latter coreference: namely, that if a speaker avoids a binding instantiation of the syntactic construction actually used then a hearer should assume, unless he has good reason, that coreference is unintended. Since there is a binding instantiation of the construction used in 39b (*i.e.* 39a) and this was avoided, coreference was not intended.

It should be clear that neither Lasnik's nor Reinhart's notion of discourse context is sufficient to handle discourse cases such as 36a,b above. The reason, of course, is that we cannot say that 'a man' and 'a donkey' *refer* to individuals and thereby introduce individuals into the discourse context. (If there were a unique donkey or man available we might be able to make use of something like Evan's definite description strategy *cf.* [Evans 77]). Similar notions of discourse context are invoked by some recent computational systems for deriving meaning representations and I discuss two of these, Candide [Pereira & Pollack 91] and the Core Language Engine [Alshawi 90], in the next chapter.

It is worth pointing out that Reinhart's theory is quite inappropriate for our purposes even for those cases with which it is intended to deal. According to Reinhart, a hearer who encounters 39b will assume non-coreference between 'Felix' and 'he', since the speaker has avoided using 39a. It is important that, so far, this has not given us a rule to determine the truth conditions of the uttered sentence but, rather, guidance as to the *intentions* of the speaker. Suppose 39b were uttered by a speaker whilst pointing at Felix. Then a hearer, who we suppose recognises Felix, will follow Reinhart's advice and assume that coreference was not intended. But he has used something else to discover what the truth conditions of the sentence were, namely a rule such as Kaplan's in [Kaplan 89] where the value of a pronoun accompanied by a demonstration is the object demonstrated. Use of this rule enables him to see that, in this instance, 39b is true if Felix thinks that Felix is a genius. The need for a rule of interpretation comes out even more clearly if we consider cases highlighted by Evans in [Evans 80]. Evans argued that the primary semantic notion used in an explanation of anaphoric dependency should not be coreference, or even *intended* coreference. Evans used a case such as 40a and Reinhart herself cites 40b to show that pronouns can both corefer and be intended to corefer with other expressions which they precede and c-command.

(40) a What do you mean John loves no one?
He loves John.
b He is Zelda's husband

These are prima-facie counterexamples to Reinhart's account because the structures used above in 40a and 40b are to be avoided if coreference is intended, unless the speaker has good reason. Reinhart's response to 40a,b is to invoke the 'good reason' clause. There is (as Evans himself pointed out) a certain *point* in using the construction 'He loves John' to express the fact that John loves himself - it is thereby emphasized that the only counter-example to the claim that no-one is loved by John is John himself. This conversational point supplies us with good reason to override the non-coreference assumption. However, the conversational point cannot help us to interpret the sentence in the first place. It is rather one consequence of our having already understood it. So

one cannot use it in an explanation of what the truth conditions of 40a are. Similarly, Reinhart's response to 40b is to claim that its binding instantiation 'Zelda's husband is himself' means something quite different from 40b ; and it is this that supplies us with good reason to override the default non-coreference assumption. But this fact can hardly be used to further *our* objective, which is to predict what the sentences mean in the first place.

Discourse Contexts in DPL

Given our concern with discourse anaphora, one can be forgiven for supposing that DPL already contains a suitable notion of discourse context. In fact, this is not unproblematically so. A context in DPL is an assignment of objects to variables. This notion differs rather sharply from the idea of a context as a perceptual environment or even an environment of objects that have been referred to. In particular, DPL contexts contain logical variables which, of course, are standardly part of the syntax of the translation language or logical form. The idea of speakers and hearers *inspecting or searching* such contexts for possible referents seems to be distinctly strange. Part of the objection of 'non-compositionality' to DRT resided in the question of the status of the Discourse Representation Structures - what *sort* of information is it that a hearer appeals to when he searches a DRS for an accessible antecedent ? Is it (just) syntactic information or does it include also some further 'mental representation' ? The same question might be posed to dynamic logic - in what *sense* is an assignment of variables to objects really a context in which one might evaluate a formula ? In the sense in which a DRS apparently encodes more than ordinary syntactic structure, do not DPL assignments do so too ?

These questions result from trying to think of the input and output contexts of DPL in a *realistic* fashion. Can one think of DPL contexts as contexts in the sense in which we think of perceptual contexts or even the discourse contexts of Lasnik and Reinhart

? I think not. But, if not, then in what sense are we to think of them?

Perhaps the notion of context in DPL is a purely theory-internal notion given its sense solely by the functional role it plays in the resulting truth-theory. In the same sense, 'satisfaction' in first order logic is not generally accorded any further importance other than its role in the definition of truth. One does not there have any worry about what an assignment *is* and how it is related to a formula : what it is and what is its role is entirely given by its definition and the axioms governing how assignments are to be used. This is a perfectly good way to think about dynamic logic but it does lose us the idea that dynamic logic shows us how 'The utterance of a sentence brings us from a certain state of information to another one' ([Groenendijk & Stokhof 91b]); at least, it does if these states of information are genuinely psychological states of a speaker or hearer. The notion of change of state of a hearer as he understands a discourse also naturally brings to mind the notion of 'incremental interpretation' - an idea that Groenendijk and Stokhof also allude to - but whose discussion I postpone to chapter 7. It is in any case quite alien to the idea that contexts and context change are purely theory-internal notions in Dynamic Logic.

A New Notion of Discourse Context

I shall take a context to be not a construction out of logical form variables and objects (namely, a function from one to the other) but out of noun phrase indices and objects. Each noun phrase in a discourse will have a unique identifier. One might take each identifier as identifying an individual *use* of a noun phrase. This notion of context will not suffer from doubts over its proper status as did the DPL context. The context will contain no logical-form variables but perfectly proper syntactic objects (unique noun phrase indices). The price to be paid, of course, is that our contexts will no longer *automatically* assign the same value to distinct occurrences of linguistic items. That is, one use of variables in logic is simply to identify the values of different argument places

to a predicate. Within the scope of a quantifier binding x, mere repetition of x ensures identical evaluation. English contains no such similar device. In this respect, one can criticise early versions of Montague Grammar which incorporated this possibility by the use of indexed pronouns. That is, there are lexical items such as he_1 which one can use to construct a sentence such as 41

(41) He_1 came in and he_1 sat down

where the two occurrences of he_1 are guaranteed to have the same denotation. A lexical choice amongst distinct pronouns has determined coreference. (Sometimes the objection is expressed that English just doesn't contain a *word* such as he_1 - but the proper force of the objection is only felt when the semantic consequences of this possible lexicon are considered *cf.* the 'well-formedness constraint' of [Partee 79]). Our noun phrase indices will therefore play the role of uniquely marking argument places but not of ensuring coreference, which is the role that indices played in the Rooth Fragment presented in section 3.6.

Let us reconsider a classic case of discourse anaphora.

(42) A man came in. He sat down

In DPL, one secures an anaphoric dependency between 'a man' and 'he' by (somehow) co-indexing the two noun-phrases and then *threading* the output assignment for the first sentence into the input assignment for the second. The hard work goes into securing the co-indexing. Once that is achieved, the only relation we need between the output of the first sentence and the input to the second is *identity.* The alternative I suggest is to remove the search for co-indexing of noun phrases across sentences and to secure the dependency between the two noun phrases by the *relation* between the contexts. For example, the output context for the first sentence in 42 will encode a unique index for 'A man' plus its assignment to an object (one which is a man). It is the semantic rule for interpreting 'He sat down' in that context which will secure the dependency

of the value of 'He' on that for 'A man'. Thus the interpretation of 'He sat down' will depend directly on the context in which it is evaluated and not on a prior co-indexing of noun phrases based on an otherwise quite unnatural syntactic notion.

One result of this is that we will be making use of an *asymmetric* relation of anaphoric dependency, rather than the purely symmetrical one of co-indexing. If the interpretation of 'He' depends on that of 'A man', it does not follow that the interpretation of 'A man' depends on that of 'he'. By contrast, if 'he' is coindexed with 'A man' then 'A man' is co-indexed with 'he'. The idea of using an asymmetric relation in handling anaphora is one which, to my knowledge, was first explicitly proposed by Evans in [Evans 80]. Higginbotham has provided a linguistically sophisticated version of Binding Theory called 'Linking Theory' which is designed to rival the Binding Theory component of Government and Binding Theory. The proposal in section 4.3 will make use of similar ideas. Linking Theory makes essential use of an asymmetric relation of anaphoric dependency. Barwise also stressed the importance of using an asymmetric relation in his pioneering essay on dynamic semantics in the context of situation semantics [Barwise 87] but this early prominence of asymmetry in dynamic logic has not, so far, been sustained in the developments of [Groenendijk & Stokhof 91b] or [Groenendijk & Stokhof 91a].

4.3 The Rooth Fragment without Coreference

In this section, I develop a positive proposal which determines what antecedents a pronoun may have using the notion of what is available in the current discourse context. The proposal is explained in the context of the Rooth Fragment (RL) defined in the previous chapter (section 3.6). Before giving the new semantic rules for that fragment, I define what an RL context is. DPL contexts are just assignments from variables to objects and familiar from the semantics of FOPL. Similarly, DPL only uses *identity* and '*differs in at most the value of x from*' as relations between assignments. The

notion of an *RL* context and the relations between them will not be so familiar and need explanation. They will, however, be motivated on linguistic grounds.

4.3.1 *RL* contexts

What a context is

First I define a sequence of objects from a domain D (or D-sequence), the length of a sequence, and a partial function $\dot{s}$ for each sequence s from positive integers P to members of D

i) if $\alpha \in D$, then $\langle \alpha \rangle$ is a D-sequence
$len(\langle \alpha \rangle) = 1$
$\dot{\langle \alpha \rangle}(1) = \alpha$

ii) if $\alpha \in D$ and s is a D-sequence, then $s \cdot \alpha$ is a D-sequence
$len(s \cdot \alpha) = len(s) + 1$
$\dot{s}(len(s) + 1) = \alpha$

We will write $\langle \alpha\beta \rangle$ for $\langle \alpha \rangle \cdot \beta$, $\langle \alpha\beta\gamma \rangle$ for $\langle \alpha\beta \rangle \cdot \gamma$ and so on. As an example, let s be $\langle \alpha\beta\gamma\delta \rangle$, then s is a D-sequence, $len(s) = 4$, $\dot{s}(1) = \alpha$, $\dot{s}(2) = \beta$, $\dot{s}(3) = \gamma$, $\dot{s}(4) = \delta$ and $\dot{s}(n)$ is undefined for other values of n. Also, $\alpha \in s$ will mean that α occurs somewhere in the sequence s *i.e.* $\exists i\ (\alpha = \dot{s}(i))$.

Let O be a domain of objects. Consider a sequence of sequences S such that the first member of each sequence is a member of O and all of whose other members are members of P *i.e.*

$$\forall_{s \in S}(\dot{s}(1) \in O \ \wedge \ \forall_n(n \neq 1 \rightarrow (\dot{s}(n) \text{ is undefined } \vee \ \dot{s}(n) \in P)))$$

If such a sequence also meets the condition that no number occurs in two sequences or twice in the same sequence, then the sequence is an *RL context*. That is, if S meets the condition

$$\sum_{s \in S}(len(s) - 1) = |\bigcup_{s \in S}\{x : x \in s,\ x \neq \dot{s}(1)\}|.$$

then S is an RL context.

Here are some example contexts where $O = \{a, b, c\}$.

a $\langle\langle a, 1\rangle\rangle$
b $\langle\langle a, 1\rangle, \langle b, 2, 3\rangle, \langle a, 4\rangle\rangle$
c $\langle\langle a, 19, 21, 57\rangle, \langle a, 20, 42\rangle, \langle a, 1, 99\rangle, \langle a, 6\rangle, \langle a, 32, 9\rangle\rangle$

The idea behind these complex objects is that the numbers that occur in the sequences will be the indices of NPs in the discourse (according to the Rooth fragment, NPs are uniquely indexed). The value assigned to an index will be given by the first object in the sequence in which that index occurs. If two indices occur in the same sequence then they will therefore receive the same value. In fact, if one index i occurs *after* another j in a sequence then the linguistic expression indexed by i is anaphorically dependent on the expression indexed by j. Each sequence within a context is thereby a record of a chain of anaphoric dependency within the discourse. We use *sequences* of sequences and not just sets of sequences just so that when we define *relations* between input and output contexts we can identify easily which member of an output context corresponds to which member of the input sequence.

Consider again the examples in 40, which I have repeated below with unique NP indices appended.

(43) a What do you mean John_1 loves no one?
He_2 loves John_3.
b He_{19} is $(\text{Zelda}_{20}\text{'s husband})_{21}$

As we discussed earlier, in 43a 'he_2' and 'John_3' can *corefer*, in the sense of both denoting John. Yet the sentence 'He_2 loves John_3' is a classic 'Condition B' case of Government and Binding Theory where one would usually conclude that they do *not* corefer. According to Evans and Higginbotham, what the 'Condition B' case shows is simply that 'he_2' cannot *anaphorically depend* on 'John_3', *i.e.* the fact that 'he_2' refers to John in no way depends on the fact that 'John_3 refers to John. Rather, 'he_2' refers to John because 'John_1' refers to John and 'he_2' depends on 'John_1.

Our contexts are designed to allow for this sort of possibility. Formally, the index 2 will occur in a different sequence within the context than that in which the index 3 occurs. It will occur in the sequence containing the index 1. But these two different

sequences will each have John as their first member. Therefore 'he_2' will refer to John but it will anaphorically depend on '$John_1$', not '$John_3$'. A similar line of reasoning applies to 43b, where 'he_{19}' will not anaphorically depend on 'Zelda's $husband_{21}$' and so 19 will not occur after 21 in a sequence. Nevertheless, 19 may occur in another sequence whose first member is, in fact, Zelda's husband.

The above examples illustrate why our contexts are composed of sequences (because they are chains of anaphoric dependency) and why different sequences can yet have the same object as their first member. Our contexts also impose the stronger condition that no index may occur twice in the same sequence or in two different sequences within a context. The former condition is justified on the grounds that no pronoun can depend on itself for its own interpretation. Otherwise, in order to determine the denotation of a pronoun one would first have to determine the denotation of that pronoun - and so no denotation would ever be defined. Consider example 44, due to [Higginbotham 83], where we illustrate chains of dependency by arrows.

(44) * [His wife] saw [her husband]

The value of 'his' depends on that of 'her husband' whose value in turn will depend on that of its constituent 'her', whose own value anaphorically depends on that of 'his wife' whose value finally depends on that of its constituent 'his'. Therefore the interpretation of 'his' depends on itself and this renders the sentence (with dependencies as illustrated) uninterpretable.

The second condition that no index may occur in two different sequences may seem too strict a condition; indeed, Higginbotham uses such a possibility to handle certain plural anaphora as in 45, where 'they' depends on both 'John' and 'Mary' but 'John' does not depend on 'Mary' nor 'Mary' on 'John'.

(45) John told Mary that they should leave

In the domain of singular anaphora, which forms the topic of this chapter, the strict condition is, I suggest, justifiable. For if 'anaphoric dependency' means 'the pronoun takes its value from that of its antecedent' then what could be meant by allowing a pronoun to take its value from several antecedents? Either those antecedents have the same value, in which case nothing is gained by letting the pronoun depend on more than one of them, or the antecedents have different values, in which case we do not have a consistent story as to what the value of the pronoun is. I discuss the issue of possible multiple dependencies involving one expression further below (section 4.4.3).

Relations between Contexts

We need to define relations between contexts to handle both quantification and our cases of discourse anaphora. To handle quantification, we define the RL analogue of 'differs at most in the value of x from'. If S is a context, $i \in P$ and $\alpha \in O$, then S_i^α is either that context just like S except where $i \in s$, $\dot{s}(1)$ is α, or else S_i^α is $S \cdot \langle \alpha, i \rangle$ (if i occurs in no sequence in S).

Secondly, we define the *extension* of a context Y by another X with respect to an index i, written $X \supset_i Y$

$$S \supset_i R \text{ iff } \exists s(s \in S \ \wedge \ i \in s) \ \wedge \ \forall_j(\dot{S}(j) = \dot{R}(j) \ \vee \ \dot{S}(j) = \dot{R}(j) \cdot i)$$

Informally, one context extends another by i just if S and R are identical except that one sequence in S has the index i added to the end. An example possible context is shown in 46a and all possible extensions with respect to 7 shown in 46b,c,d.

(46) a $\langle\langle a, 1\rangle, \langle b, 2, 3\rangle, \langle a, 4\rangle\rangle$
b $\langle\langle a, 1, 7\rangle, \langle b, 2, 3\rangle, \langle a, 4\rangle\rangle$
c $\langle\langle a, 1\rangle, \langle b, 2, 3, 7\rangle, \langle a, 4\rangle\rangle$
d $\langle\langle a, 1\rangle, \langle b, 2, 3\rangle, \langle a, 4, 7\rangle\rangle$

$\langle\langle a, 1, 7\rangle, \langle b, 2, 3\rangle, \langle a, 4, 7\rangle\rangle$ is not an extension of 46a simply because it is not a context at all. The number 7 occurs twice. The relation of extension between contexts is designed to correspond to adding an NP index into a chain of anaphoric dependency.

Syntactic constraints on the adding of indices into chains (for example, concerning the distinction between reflexive and non-reflexive pronouns) will be considered later.

4.3.2 *RL*+ Semantic Rules

In this section we re-interpret the language *RL* using the notion of context developed above. The result is called *RL*+.

RL+ models are identical to those for *RL*. The interpretation for *RL*+, unlike that of *RL*, is dynamic because we define satisfaction of a formula with respect to pairs of input contexts and output contexts. We will also define the value of terms relative to input and output contexts - because when we interpret a pronoun, we shall want to add its index into a chain of dependency. The relation **R** holds between objects α, indices x, input contexts i and output contexts o as follows

$\mathbf{R}(\alpha, x, i, i)$ *if* $\exists s(s \in i \ \wedge \ x \in s \ \wedge \ \dot{s}(1) = \alpha)$
$\mathbf{R}(\alpha, x, i, o)$ *if* $o \supset_x i \ \wedge \ \mathbf{R}(\alpha, x, o, o)$

That is, if an index is already in a chain of dependency then its value is the first member of the chain; otherwise we add the index into a chain and then take the first member of that chain.

Quantifier Rules

1 $[\![\Sigma([_{NP^x}\ every_{DET}\ \delta_{N^x}])]\!]^{h\ k} = 1$ if $h = k \wedge \forall\alpha\ \forall l\ ([\![\delta_{N^x}]\!]^{h^\alpha_x\ l} = 1 \rightarrow \exists m[\![\Sigma[it_{NP^x}/\Lambda_{NP^x}]]\!]^{l\ m} = 1)$

2 $[\![\Sigma([_{NP^x}\ a_{DET}\ \delta_{N^x}])]\!]^{h\ k} = 1$ if $\exists\alpha\ \exists l\ ([\![\delta_{N^x}]\!]^{h^\alpha_x\ l} = 1 \wedge [\![\Sigma[it_{NP^x}/\Lambda_{NP^x}]]\!]^{l\ k} = 1)$

Structural Rules

3 $[\![[_S\ it_x\ \delta_{VP}]]\!]^{h\ k} = 1$ if $\exists\alpha\exists j(\mathbf{R}(\alpha, x, h, j) \wedge \alpha \in [\![\delta_{VP}]\!]^{j\ k})$

4 $[\![[_{N^x}\ v_{N^x}\ \delta_{PP^y}]]\!]^{h\ k} = 1$ if $\exists j([\![v_{N^x}]\!]^{h\ j} = 1 \wedge [\![\delta_{PP^y}]\!]^{j\ k} = 1)$

5 $[\![[_{PP^x}\ \delta_P\ [_{NP^y}\ it]]]\!]^{h\ k} = 1$ if $\exists m\exists\alpha\exists\beta(\mathbf{R}(\alpha, x, h, m) \wedge \mathbf{R}(\beta, y, m, k) \wedge \langle\alpha, \beta\rangle \in F(\delta_P))$

6 $[\![[_{VP}\ \delta_V\ [_{NP^x}\ it]]]\!]^{h\ k} = \{y \mid \exists\alpha(\mathbf{R}(\alpha, x, h, k) \wedge \langle y, \alpha\rangle \in F(\delta_V))\}$

Lexical Insertion Rule

7 $[\![\delta_{N^x}]\!]^{h\ k} = 1$ if $\exists\alpha(\mathbf{R}(\alpha, x, h, k) \wedge \alpha \in F(\delta_N))$

Discourse Rules

8 $[\![[_D\ \delta_S\ v_D]]\!]^{h\ k} = 1$ if $\exists l\ ([\![\delta_S]\!]^{h\ l} = 1 \wedge [\![v_D]\!]^{l\ k} = 1)$

9 $[\![[_D\ \delta_S]]\!]^{h\ k} = 1$ if $[\![\delta_S]\!]^{h\ k} = 1$

Ex No formula is assigned a semantic value except in virtue of the above rules.

An *RL+* discourse, ϕ is *true* just in case $\exists p\ ([\![\phi]\!]^{\langle\rangle\ p} = 1)$.

The dynamic elements in the above definitions come in two places. First, there is the threading of contexts across formulae which occurs in the two quantification rules (1 and 2), the complex noun rule (4) and the discourse rule (8). This form of threading is familiar from DPL (see 2.2 above). Secondly, there is threading within an atomic formula as particular pronoun occurrences are processed. There are alternatives to this latter procedure. One could remain with the idea of just threading at formula level by defining the class of all free pronouns for an atomic formula and have a more complex notion of when one context extends another which could add sets of indices into a context. However, I think the idea of threading across the interpretation of terms is a clearer one and makes plain that terms can have a relational interpretation too.

It should be noted that the above quantifier rules use the 'outside-in' treatment of the previous chapter but quantifiers are not treated as binary structured operators as they were earlier. The meta-language quantifiers used on the right-hand-side of the rules are unary operators only, just like in DPL. The question of how to treat binary structured quantifiers within dynamic logic forms the topic of chapter 6 of this thesis. However, I shall make some brief remarks here. One popular way to incorporate them (following suggestions by [Chierchia 88] and [van Eijck & de Vries 91]) is to adopt the following sort of interpretation rules

$$[\![Qx\ (A; B)]\!]^{i\ o} \text{ iff } i = o\ \wedge\ Q'\alpha\ (\exists g\ [\![A]\!]^{i^{\alpha}_{x}\ g}\ ;\ \exists h\ ([\![A]\!]^{i^{\alpha}_{x}\ h}\ \wedge\ \exists j\ ([\![B]\!]^{h\ j})))$$

The right-hand-sides of these interpretation rules do use binary structured operators. However, three points should be noted about these rules. First, the evaluation of A appears *twice* in such a rule. Now the leading idea in dynamic semantics is that the output context resulting from one evaluation forms the input context for the next evaluation. It therefore seems a little strange that in the above rules one evaluates A within the first argument to the binary structured quantifier and then one evaluates it once again within the second argument. If one wants information derived from evaluating the first argument to reach the second argument, one would expect, in a dynamic environment, to be able to thread information from one to the other. In these rules however, one has to evaluate an expression a second time in order to achieve this. In chapter 6, I will propose a new account which does thread the output context from the first argument of a binary structured quantifier into the second argument.

The second point to note about these rules is that they do not give quite the same truth conditions as the rules of DPL. In particular, the DPL translation of 'every N VP', $\forall x(N' \rightarrow VP')$, means that the output contexts for N are universally quantified over whereas a translation using the above schema $\forall x(N'\ ;\ VP')$ only uses existential quantification over the output contexts for N. There is considerable debate over which truth conditions are the correct ones and I again refer the reader to chapter 6 for further

discussion of this. The earlier rules are those which provide predictions in agreement with our exposition of DRT (2.1).

The final point to note about the 'copying' of material within binary structured quantifiers is that this strategy will give unwanted results if one is copying ambiguous surface material. Chierchia's and van Eijck's rules work because they are defining semantics for disambiguated formal languages. Our project however is to define a dynamic semantics for a natural language. Suppose, for example, that we followed their strategy and evaluated the noun which restricts the quantified noun phrase (the first argument to the quantifier) twice - first, to give the restriction to the quantifier and secondly together with the verb-phrase to allow anaphoric bindings between the two. Suppose the material to be evaluated twice were 'boy that saw a man that chased a thief that robbed him' where 'him' could refer either to the boy or the man. Then the first evaluation could result in one meaning and the second in the other. This would result in incorrect truth conditions. For example, one meaning for 'every boy that saw a man that chased a thief that robbed him cheered loudly' would be that every boy that saw a man that chased a thief that robbed that man was a boy who both saw a man that chased a thief that robbed that boy and cheered loudly'!

In the light of these three points, I will continue to use unary quantifiers for the remainder of this chapter and return to the analysis of binary structured quantifiers in chapter 6.

Simple Example

(47) a A man saw a boy. He greeted him.

b $[_D\ [_S\ [_{NP_1}\ \mathrm{A}_{DET}\ \mathrm{man}_{N^1}]\ [_{VP}\ \mathrm{saw}_V\ [_{NP_2}\ \mathrm{a}_{DET}\ \mathrm{boy}_{N^2}]]]$
$[_D\ [_S\ \mathrm{He}_{NP^3}\ [_{VP}\ \mathrm{greeted}_V\ \mathrm{him}_{NP^4}]]]]$

The discourse begins with an input context of $\langle\rangle$. The overall discourse structure is $[_D\ S\ D]$ so, by rules 8 and 9, the output context for the first sentence (call this p) is threaded into the second sentence. We will first derive the output context for the

first sentence. Since the derivation will proceed top-down through the structure 47b, at each stage I shall only indicate the top-most syntactic structure.

Consider then, $[_S\ [_{NP_1}$ A man$]\ [_{VP}$ saw a boy $]]$. First we apply the rule for truth:

$$\exists p\ [\![\ [_S\ [_{NP^1}\ \text{A man}]\ [_{VP}\ \text{saw a boy}]]\ ]\!]^{\langle\rangle\ p} = 1 \text{ if } \ldots$$

There are now two possible rule-applications here (for 'a man' and 'a boy'): choose 'a man'.

a) [2] $\exists\alpha\ \exists m\ ([\![\ \text{man}_{N^1}\ ]\!]^{\langle\langle\alpha,1\rangle\rangle\ m} = 1 \wedge [\![\ [_S\ \text{it}_{NP^1}\ [_{VP}\ \text{saw a boy}]]\ ]\!]^{m\ p} = 1)$

When we evaluate $[\![\ \text{man}_{N^1}\ ]\!]^{\langle\langle\alpha,1\rangle\rangle\ m}$, we find rule 7 tells us to find an object γ and output context k such that $\mathbf{R}(\gamma, 1, \langle\langle\alpha,1\rangle\rangle, k)$. By definition of $\mathbf{R}$ γ is just α and k is $\langle\langle\alpha,1\rangle\rangle$. Then we need to check that $\gamma \in F(man)$. The result of this is

b) [7] $\exists\alpha\ (F(\text{man})\alpha \wedge [\![\ [_S\ \text{it}_{NP^1}\ [_{VP}\ \text{saw a boy}]]\ ]\!]^{\langle\langle 1,\alpha\rangle\rangle\ p} = 1)$

Now we choose to use rule 2 again and 'a boy'.

d) [2] $\exists\alpha\ (F(\text{man})\alpha \wedge$
$\exists\beta\ \exists n\ ([\![\ boy_{N^2}\ ]\!]^{\langle\langle\alpha,1\rangle\langle\beta,2\rangle\rangle\ n} = 1 \wedge [\![\ [_S\ \text{it}_{NP^1}\ [_{VP}\ \text{saw}_V\ \text{it}_{NP^2}]]\ ]\!]^{n\ p} = 1))$

By similar steps to those just employed, we find

e) [5] $\exists\alpha\ (F(\text{man})\alpha \wedge$
$\exists\beta\ (F(\text{boy})\beta \wedge$
$[\![\ [_S\ \text{it}_{NP^1}\ [_{VP}\ \text{saw}_V\ \text{it}_{NP^2}]]\ ]\!]^{\langle\langle\alpha,1\rangle\langle\beta,2\rangle\rangle\ p} = 1))$

Finally, by rules 3 and 6, we find that $[\![\ [_S\ \text{it}_{NP^1}\ [_{VP}\ \text{saw}_V\ \text{it}_{NP^2}]]\ ]\!]^{\langle\langle\alpha,1\rangle\langle\beta,2\rangle\rangle\ p} = 1$ if $\alpha \in \{y \mid \langle y, \beta\rangle \in F(saw)\}$,and that p is $\langle\langle\alpha, 1\rangle\langle\beta, 2\rangle\rangle$.

f) [3,6] $\exists\alpha\ (F(\text{man})\alpha \wedge$
$\exists\beta\ (F(\text{boy})\beta \wedge$
$F(\text{saw})(\alpha, \beta)$

So far, the derivation has proceeded just as it might in ordinary dynamic predicate logic - we have simply reassured ourselves that the extra complexity in finding the values of terms has not altered our treatment for the more basic cases. We have derived an appropriate truth condition for 'A man saw a boy' and we have an output context p to thread into the next sentence: $\langle\langle\alpha, 1\rangle, \langle\beta, 2\rangle\rangle$.

Now we evaluate the second sentence, $[_S\ \text{He}_{NP^3}\ [_{VP}\ \text{greeted}_V\ \text{him}_{NP^4}]]$, with an input context $\langle\langle\alpha,1\rangle,\langle\beta,2\rangle\rangle$. We want to know (where q is an output context for the whole sentence)

$$[\![\ [_S\ \text{He}_{NP^3}\ [_{VP}\ \text{greeted}_V\ \text{him}_{NP^4}]]\ ]\!]^{\langle\langle\alpha,1\rangle,\langle\beta,2\rangle\rangle\ q} = 1 \text{ if } \ldots$$

Rule 3 is again applicable and at this point the second clause of our definition of **R** is important, because rule 3 tells us to find an object γ and an output context j such that $\mathbf{R}(\gamma,3,\langle\langle\alpha,1\rangle,\langle\beta,2\rangle\rangle,j)$ but 3 is not in the input context. Therefore we extend the input context to another one where it is present. There are two choices for this

i) $\langle\langle\alpha,1,3\rangle,\langle\beta,2\rangle\rangle$
ii) $\langle\langle\alpha,1\rangle,\langle\beta,2,3\rangle\rangle$

If we choose i), then the value of He_3 will be α. Then we must next evaluate

$$[\![\ [_{VP}\ \text{greeted}_V\ \text{him}_{NP^4}]\ ]\!]^{\langle\langle\alpha,1,3\rangle,\langle\beta,2\rangle\rangle\ q}$$

which, by rule 6, will give us another choice of which sequence to add the index 4 to. There is nothing, so far, to prevent us adding to the same sequence that we added 3 to. Therefore the value of 4 and 3 may be the same the object assigned to 1, or 2. Suppose we choose to add 4 to the sequence $\langle\beta,2\rangle$, then the value of him_4 will be β. So 'He_3 greeted him_4' will be true if $\alpha \in \{y \mid \langle y,\beta\rangle \in F(\mathit{greeted})\}$, that is if $F(\mathit{greeted})(\alpha,\beta)$. The output context q will be also be threaded out and is $\langle\langle\alpha,1,3\rangle,\langle\beta,2,4\rangle\rangle$.

Overall, the truth condition for the discourse will be that a man saw a boy and that that man greeted that boy. Since we could have chosen to add the indices 3 and 4 into the context differently, the sentence also has readings where a man saw a boy and where the boy greeted the man, where a man saw a boy and greeted himself and where a boy was seen by a man and greeted himself.

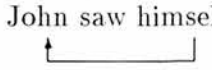
The conclusion is that, modulo the necessary addition of some syntactic constraints to rule out cases such as 'he' and 'him' co-referring in our example, the example has the two truth conditions that we expect. Most importantly, the derivation has not depended on a prior co-indexing of noun phrases but has used the dynamic threading

of unique noun phrase indices in order to predict the possible anaphoric dependencies.

4.4 Syntactic Constraints on Pronominal Dependencies

In this section, I modify the notion of 'extension of one context by another' to incorporate syntactic constraints on possible anaphora. That is, I alter the definition of $\supset_i$ as used in the semantics for $RL+$. Since our contexts include an asymmetric relation of dependency between anaphoric pronouns and their antecedents, I shall start by discussing a simple version of Higginbotham's Linking Theory, which is perhaps the most linguistically sophisticated theory to use such an asymmetric relation.

A *link* can be pictured as an arrow leading from the dependent anaphor to its (immediate) antecedent, like so



The transitive closure of the linking relation is called *antecedence*.

Higginbotham is not entirely explicit about what a link is (I discuss this further below) but I shall take it that 'x is linked to y' means what 'x is the immediate successor of y in a sequence within a context' means in $RL+$. Evans used a notion called 'referential dependency' in his [Evans 80] explaining it as one term 'picking up its reference' from another, which also appears to be the same concept. If a pronoun is linked to a proper name then its value is that of the proper name; and if a pronoun is linked to a quantified noun phrase, then it behaves as a variable bound by the relevant quantifier. Antecedence is now that relation which holds between two indices when one occurs *somewhere* before the other in a sequence within a context.

Higginbotham's Linking Theory makes use of five main principles, which I shall state first before providing some explanation for them.

GC If A c-commands B, then B is not an antecedent of A
A If A is an anaphor, there is exactly one B in G(A) such that B c-commands A and A is linked to B
B If A is pronominal and B c-commands A in G(A), B is not an antecedent of A
C A full-referring expression is not to be linked
FC If A and B share an antecedent and A c-commands B, then A is an antecedent of B

Condition **GC** (or General Condition) uses the notion of antecedence, introduced by Higginbotham as the transitive closure of 'linking'. Conditions **A**, **B** and **C** (so called from their correlates in Binding Theory [Chomsky 81]) use the terms 'anaphor', 'pronominal' and 'full referring expression' meaning, for our purposes, 'reflexive pronoun', 'non-reflexive pronoun' and 'individual denoting expression whose evaluation depends on no other expression'. G(A) means the 'governing category' of A which, again for our purposes, we can take to be the smallest clause containing a verb and the expression A. The name **FC** simply stands for Further Condition.

Conditions **A**, **B** and **C** receive similar justifications to their Binding Theory counterparts as witnessed in the following acceptability judgments (48a,b for **A**; 48c,d for **B**; 48e for **C**)

(48) a John saw himself
b * John believed Mary loved himself
c * John saw him
d John believed Mary loved him
e * He saw John

GC is needed to account for the following data

(49) * He saw him

Higginbotham motivates **FC** on the grounds that the other principles are insufficient to predict the following judgment

(50) * John said he saw him

According to **FC** however, 'he' is an antecedent of 'him' and so 50 violates condition **B** after all. Clearly, this example is a parallel one to our own earlier worked example 47 and is therefore of some importance for discourse anaphora. Higginbotham does not concern himself with extra-sentential cases. In $RL+$ however, we will have no use for **FC** because RL contexts do not allow the possibility that two indices are both *linked* to a third index. Each index can occur in only one sequence within a context and not twice within a context. I discuss this issue further below.

4.4.1 Syntactic Constraints in the Rooth Fragment

Rather than directly add Linking Theory to $RL+$, I shall use a similar account using notions of *locality* and *precedence*. The main justifications for this are, first, that the resulting theory is somewhat simpler and, secondly, that a locality based constraint makes more sense when we come to develop, as we shall, a Hobbs and Shieber style algorithm which takes 'predicate-argument' relations as input. Furthermore, there are linguistic arguments indicating that a locality based account is superior.

[Partee & Bach 80] contains an interesting discussion of the relative merits of configurational properties of trees versus function-argument structure in anaphora resolution. They conclude that an account based on the latter notion is at least as good as one based on the former. For example, instead of saying that a pronoun cannot co-refer with a nounphrase that it c-commands, one can say that a pronoun cannot co-refer with

a nounphrase in a constituent which is a function having the pronoun as argument. This rules out

(51) * He loves John's mother

because 'he' is an argument to the function 'loves John's mother' but that function contains the proposed antecedent 'John' as a constituent. In some cases, the use of function-argument structure proves superior to a c-command account . In 'I talked to him about John', 'him' does not c-command 'John' and therefore co-reference ought to be possible according to a typical Binding Theory account. If we assume that the preposition 'about' serves simply to mark an argument position of 'talk' and has no semantic content itself, then 'him' and 'John' are both local arguments of 'talk' and so a function-argument structure account will correctly predict that they may not co-refer.

To incorporate locality and precedence into Linking Theory, I adopt the following versions of Higginbotham's principles.

GC' If A precedes B, then A is not linked to B
A' If A is an anaphor, there is exactly one *local* B such that B precedes A and A is linked to B
B' If A is pronominal and B is local to A , B is not an antecedent of A
C' A full-referring expression is not to be linked

I include precedence in the conditions largely because I am not going to consider the possibility of cataphora. In general, cataphora presents an at least prima-facie problem for dynamic theories because pronouns are evaluated on the basis of what has already been interpreted, whereas the cases of cataphora are precisely those where the antecedent of a pronoun *follows*, at least temporally, the dependent pronoun. It is of course true that we are allowing for quantifier scope ambiguities where dependencies are also the reverse of temporal order. However, the inclusion of cataphora would also force us to distinguish between the ways in which pronouns relate to quantified noun phrases and proper names, witness

* His mother loves John
His mother loves every man

That is, in systems employing a version of quantifier raising, great care needs to be taken with 'weak crossover' cases such as 'His mother loves every man' where even if 'every man' takes scope over the whole sentence, it cannot bind the pronoun 'His'. In contrast, if we quantify 'John' in at sentence level then it *can* bind the pronoun. I have chosen to divide the data by reserving cataphora for some other mechanism and thereby preserve the similarities between the binding potential of quantified noun phrases and proper names.

Higginbotham's condition **FC** is om itted because, as I indicated earlier, $RL+$ contexts are set up so that no item can occur in more than one chain of dependency and it is one of these sorts of possibility that **FC** is designed to eliminate in the first place.

Given the simple syntactic coverage of the Rooth Fragment, the question of what locality is is a simple matter. The only complex case is the PP rule which is designed to avoid complications with extraction sites . In this instance, I simply stipulate that the object noun phrase in a relative clause is local to the noun phrase which the relative clause restricts. In a fuller treatment one might want to say the object noun phrase is local to the clause's subject which is bound to the restricted noun phrase as a trace.

Formally, the definition of $\mathbf{R}(\alpha, x, i, o)$ now becomes

$\mathbf{R}(\alpha, x, i, i)$ *if* $\exists s(s \in i \ \wedge \ x \in s \ \wedge \ \dot{s}(1) = \alpha)$
$\mathbf{R}(\alpha, x, i, o)$ *if* $o \supset_{x,\phi} i \ \wedge \ \mathbf{R}(\alpha, x, o, o)$

where $o \supset_{x,\phi} i$ is a relation extending a context i to another o with respect to an index x and with respect to syntactic relationships in the discourse ϕ. I introduce two pieces of notation: where $o \supset_x i$, let o_x be that $s \in o$ such that $x \in s$ (there will only be one such). Also, let $\dot{s}^{-1}$ be the inverse of $\dot{s}$ (such an inverse exists because $\dot{s}(x)$ and $\dot{s}(y)$ are always distinct for distinct x and y).

$o \supset_{x,\phi} i$ iff a) $o \supset_x i$ and
b) $a \cdot x = o_x \ \rightarrow$ a precedes x in ϕ and
c) $a \cdot x = o_x$ & x is reflexive $\rightarrow$ a is local to x in ϕ and
d) x is non-reflexive & $\dot{o_x}^{-1}(y) < \dot{o_x}^{-1}(x) \ \rightarrow$ y is not local to x in ϕ

The new definition of when one context extends another is therefore just like the old one except in three ways: b) tells us that when an index is added into a context all the earlier items in the same sequence must precede it, c) tells us that a reflexive must be linked to a local antecedent and d) tells us that no local antecedents must exist in a sequence into which a non-reflexive is added.

The formal definition above does not include Higginbotham's condition **C'** that a full referring expression is not to be linked. This is because the Rooth fragment does not contain any such expressions, not even proper names.

Interestingly, DPL also does not contain any individual constants. In general, proper names need a distinctive treatment in dynamic semantics just because the ability of subsequent pronouns to corefer with a proper name seems quite independent of the structure of the discourse. In DRT, for example, the discourse referents for proper names are placed in the 'top-most' DRS so that no matter how the discourse is structured internally, they will always be in a DRS which all other DRSs are subordinate to. Therefore those discourse referents will always be accessible. One way to mimic this in dynamic logic is to treat proper names as effectively discourse-wide quantifiers. That is, when interpreting a discourse one first seeks out all occurrences of proper names (pn_i) and then expands the initial context $\langle\rangle$ to $\langle\langle[\![pn_1]\!] pn_1\rangle \ldots \langle[\![pn_n]\!] pn_n\rangle \ldots\rangle$. If this procedure seems odd it is because it destroys the prospect of *incremental interpretation*, even at the level of the sentence. One will first have to find all the proper names in a discourse before one can start interpreting it. Although I believe this is a good objection, I do not believe the proposal is actually not dynamic or even non-compositional - just as interpreting a quantified noun phrase in object position in English first can be both dynamic and compositional. Rather, what is brought to the fore is the need to examine the relation between dynamic interpretation and incremental interpretation.

One could try to avoid the treatment of proper names as discourse-wide quantifiers by suitably complicating one's notion of context - for example, by reserving a portion of

context to be used by proper names. That part of the context could be threaded on into all subsequent evaluations no matter what structures are encountered. However, whilst this would be an attractive manoeuvre, it really only postpones the question of whether dynamic interpretation ought to be incremental or not. The question simply arises again with respect to sub-sentential units - why are they not treated incrementally too if incremental interpretation is so closely linked to the notion of dynamic interpretation?

The issue of incremental interpretation and dynamic interpretation receives further consideration in chapter 7.

4.4.2 Worked Examples

Example 1

The first example is a re-working of 47 repeated below as 52.

(52) a A man saw a boy. He greeted him.
b $[_D\ [_S\ [_{NP1}$ A man] saw $[_{NP2}$ a boy]]] $[_D\ [_{NP3}$ He] greeted $[_{NP4}$ him]]]

The derivation for 'A man saw a boy' proceeds as before resulting in an output context of $\langle\langle\alpha\ 1\rangle,\langle\beta\ 2\rangle\rangle$. Similarly when we evaluate $[_S\ [_{NP3}$ He] greeted $[_{NP4}$ him]] in this context and use rule 3, we can add 3 into either of the two sequences in the context. However, when we use rule 6 to add 4 into context we will not be able to add it into the same sequence as 3 occurs in. Otherwise 3 would be a local, preceding antecedent of the non-reflexive, pronominal 4 which is a Condition **B'** violation. Therefore 'He_3' and 'him_4' will not co-refer in this example.

Example 2

The second example is to show how the traditional 'donkey-sentence' is handled by the new mechanism.

(53) a Every farmer with a donkey feeds it.
b $[_S\ \ [_{NP1}\ \text{Every}_{DET}\ [_{N1}\ \text{farmer}_{N_1}\ [_{PP}\ \text{with}_P\ [_{NP2}\ \text{a}_{DET}\ \text{donkey}_{N2}]]]]$
$[_{VP}\ \text{feeds}_V\ \text{it}_{NP3}]]$

As before, we shall proceed top-down through the truth-definition illustrating only the top-most syntactic structure at each stage.

[1] $[\![[_S\ [_{NP^1}\ \text{Every}_{DET}\ [_{N^1}\ \text{farmer with a donkey}]]]\ [_{VP}\ \text{feeds it}]]\!]^{\langle\rangle\ o}_{T_2} = 1$ if
$\langle\rangle = o \wedge$
$\forall\alpha\ \forall l\ ([\![[_{N^1}\ \text{farmer with a donkey}]\!]^{\langle\langle\alpha,1\rangle\rangle\ l}_{T_2} = 1 \rightarrow$
$\exists m [\![[_S\ \text{it}_{NP^1}\ [_{VP}\ \text{feeds it}]]\!]^{l\ m}_{T_1} = 1)$

Rather than give a full derivation for 'farmer with a donkey', I simply state that the phrase is true with respect to $\langle\langle \prec, l \rangle\rangle$ and l if

$$farmer(\alpha) \wedge \exists\beta\ (donkey(\beta) \wedge with(\alpha,\beta))) \wedge l = \langle\langle\alpha,1\rangle\langle\beta,2\rangle\rangle$$

Next we evaluate $[_S\ \text{it}_{NP^1}\ \text{feeds it}_{NP^3}]$ in the context $\langle\langle\alpha,1\rangle\langle\beta,2\rangle\rangle$. Rule 3 will tell us first that the value of 'it_1 is α (since 1 is already in the input context). Rule 6 will invite us to expand the input context to include the new index 3. However, we will not be able to add this into the sequence $\langle\alpha,1\rangle$ because 1 is a local, preceding index. Therefore, there is only one possibility for resolving 'it_3' in this discourse and that is to expand the context to $\langle\langle\alpha,1\rangle\langle\beta,2,3\rangle\rangle$. The value of 3 is then β. The net result for 53 is therefore

$$\forall\alpha\ \forall\beta\ ((man(\alpha) \wedge donkey(\beta) \wedge with(\beta)(\alpha)) \rightarrow feeds(\beta)(\alpha))$$

Example 3

The third example is to show how relative quantifier scopes and anaphora *interact.*

(54) a Every farmer fed a donkey. It overate.
b $[_D\ [_S\ [_{NP^1}\ \text{Every}_{DET}\ [_{N^1}\ \text{farmer}_{N_1}\ [_{VP}\ \text{fed}_V\ [_{NP^2}\ \text{a}_{DET}\ \text{donkey}_{N^2}]]]]$
$[_D\ [_S\ \text{It}_{NP^3}\ \text{overate}_{VP}]]]$

The treatment of quantified noun phrases in $RL+$ follows that of the previous chapter, therefore the analysis of 'Every farmer fed a donkey' is able to follow two distinct paths. The choice of which noun phrase to analyse first determines whether 'every' outscopes 'a' or *vice versa.* Now that we are incorporating dynamic effects too, there is an additional consequence. The output context from a universally quantified formula

differs from that of an existentially quantified formula. So the output context of 'every farmer fed a donkey' also differs depending on which quantifier noun phrase is analysed first. If 'every farmer' receives widest scope, then the output context for the first sentence will be just the input context for that first sentence, *i.e.* $\langle\rangle$ (when 54a is discourse-initial). Therefore the input context for the *second* sentence will also be $\langle\rangle$. We will not then be able to complete an analysis for the discourse because there will be no suitable candidate index for it_{NP3} in the input context. In contrast, if we choose 'a donkey' first, then there *will* be a suitable index available, namely 2.

4.4.3 Discussion of Higginbotham's rule FC

Higginbotham invoked an extra principle, **FC**, to deal with cases like Example 1, which stipulated that if two expressions share an antecedent and one c-commands the other then the former is an antecedent of the latter. The reason for this was that it was otherwise possible for two non-reflexive pronouns independently to have the same antecedent, and therefore corefer, even though one c-commanded the other. No such extra principle was required on our account simply because it is not possible for two indexes to have independently the same antecedent. If two indices share one and the same antecedent, then both those indices occur in a sequence after that antecedent. Therefore, one of those indices must be an antecedent of the other and so the two indices do not *independently* have the same antecedent.

The question is: Is there good reason to prefer our account of the matter?

There are, I think, two reasons in favour of the approach adopted here. One is on grounds of simplicity - the problematic cases are essentially classic cases to be ruled out by Condition B but they can be 'let in' by the back door if one allows the possibility of two indices both *sharing* some third antecedent. Higginbotham's condition **FC** is simply designed to rule out these back-door cases. Unfortunately, **FC** *alters* our conception of what the relation of antecedence is. The notion of 'linking' is a very natural

one and so is the notion of 'antecedence' when it is introduced just as the transitive closure of 'linking'. But principle **FC** forces us to abandon the simple intuition and tells us that one index can be an antecedent of another even if there is no chain of links between them. **FC** therefore lacks the original intuitive motivation for 'antecedence'. Our approach remains faithful to the intuitive notion.

It might be contended that Higginbotham is concerned to allow the *general* possibility of indices occurring in more than one chain of anaphoric dependency. For example, in [Higginbotham 83], one primary motivation for Linking Theory is the possibility of an analysis of pronouns with 'split antecedents' as in 55

(55) John told Mary that they should leave

Higginbotham points out that one simply could not employ *co-indexation* here with the semantic import of co-reference because 'John' does not co-refer with 'Mary' and neither does 'they' co-refer with either of them. (There is, however, the possibility of indexing noun phrases with sets of indices with the semantic import of, say, group inclusion).

However, examples like this are not in fact cases in point for they are cases where one pronoun has several antecedents whereas the putative dependencies we are trying to rule out are cases where one antecedent has several different dependents (none of which depend on each other). It is however extremely difficult to find such a distinguishing case. Indeed, according to Higginbotham's rules there is only one case where one *has* to say that two indices are independent (where one cannot postulate a linking relation between them) and this is precisely the Condition B case where a pronominal c-commands a noun phrase, just as in 'He greeted him'. The distinguishing case would therefore be one which *would* allow Condition B to be broken via the back door. Since **FC** is designed to rule out such cases, we might as well simply not permit one antecedent to have independent dependencies in the first place.

In fact, [Higginbotham 85] proposes the following putative counter-example to **FC**

(56) They told each other they should leave

If this example stands, it is a counterexample both to Higginbotham's Linking Theory and our own more restricted version. Higginbotham's idea is that if the second 'they' were linked to 'each other' then this would be similar to the control structure 'They told each other to leave' which unambiguously means that each told the other that the other should leave. 56 however also has readings where each tells the other 'I should leave' and each tells the other 'We should leave'. These latter readings are to be captured by linking the second occurrence of 'they' to the first. In the same way, 57 is two ways ambiguous.

(57) They said they should leave

We have therefore linked both 'each other' and the second occurrence of 'they' to the first occurrence of 'they' in 56. The reciprocal 'each other' cannot be linked to the second 'they' by Principle **GC**). So 56 now has two pronouns *linked* to one antecedent with neither linked to the other. As we have indicated, in as much as 56 supports the idea of antecedents having multiple dependents, it also casts doubt on Higginbotham's principle **FC**.

The counter-example depends critically both on including an analysis of plurals, which we are not considering here, and also on treating the reciprocal 'each other' simply as an anaphoric device which is linked to the initial 'they', rather like 'They told *themselves* they should leave' except for the special reciprocal effect induced by 'each other'. Whether this analysis holds good depends rather on the details of the semantic account of 'each other' which is unspecified in [Higginbotham 85]. For example, one might want the analysis of 56 to be related to that of 58

(58) Jack told the other man he should leave

However, it is not entirely obvious that 'the other man' should be *linked* to 'Jack' - for the notion of a link has still been intimately tied to that of co-reference (in the singular domain, at least). The semantic import of a link has been that the two linked indices co-refer, although the absence of a link has not entailed non-coreference. In 58 however, a link would certainly not entail co-reference. Even in the plural cases, one most naturally thinks of linking as a means of ensuring *overlap* or *inclusion* amongst objects referred to, and yet this initial impression is again disconfirmed if one allows the sort of linking seen in 56.

Of course, I am not asserting that there is no sort of *informational* dependency between 'they' and 'each other' in 56 or 'Jack' and 'the other man' in 58; but simply suggesting that whatever it is takes us some way beyond the original understanding of the notion of linking. It remains the case, however, that an adequate analysis will have to deal with plural data such as 55 at some stage and that the concept of a context as developed in this chapter is not going to be good enough to deal with that data itself.

4.5 Conclusion

A syntax and a semantics for the Rooth fragment has been defined which uses dynamic semantics in order to treat inter-sentential ('donkey') anaphora and extra-sentential anaphora and does not either make appeal to 'discourse representation structures' of uncertain status or to contexts containing logical form variables. We have also been able to integrate naturally syntactic constraints familiar from the literature in Binding Theory and not find ourselves using radically different or unnatural syntactic notions. The latter problem appears to be the inevitable result of attempting to determine purely syntactically why 59a is unacceptable and 59b is not.

(59) a * Every man who owns every donkey feeds it
b Every man who owns a donkey feeds it

Furthermore, since the issues of relative quantifier scoping and pronominal dependency interact, such an account would appear to be impossible unless quantifier scoping is

itself represented syntactically.

The alternative I have advocated is that the issues of pronominal dependency are not simply to be settled *in advance* by syntax but, as in DRT, the question of whether a pronoun has a particular antecedent is to be settled ***whilst*** we are in the interpretation process. Furthermore, as in DRT, the answer will depend partly on what the current context is (what has been interpreted so far, and how much of what has been interpreted has been passed on and into the current context) and partly on syntactic grounds. In DPL, as in most standard semantic theories in the logical tradition, one appeals to syntax only in order to determine *which rule to apply next.* Other issues, such as pronoun resolution and quantifier scoping, are assumed to have been sorted out already. In contrast, in $RL+$, the rules appeal to syntactic relations during the process of interpretation. It is a particularly attractive feature of $RL+$ that these rules have been lifted from the existing linguistic literature and yet they are sufficient to handle the discourse anaphora in which we are interested.

Chapter 5

A Dynamic Algorithm

5.1 Introduction

In this chapter I shall first describe an algorithm based on the semantic theory for $RL+$ presented in the previous chapter. Secondly, I shall discuss some other approaches to similar 'dynamic phenomena' in the recent computational linguistics literature. The discussion will include a proposal of Latecki and Pinkal's in the framework of DRT [Latecki & Pinkal 90], the system Candide [Pereira & Pollack 91] and the SRI Core Language Engine [Alshawi 90]. The latter two systems not only attempt to handle the phenomena but also invoke the idea of discourse models being threaded through the evaluation of a discourse in order to handle 'contextual influences'.

5.2 An $RL+$ Algorithm

The input data structures will, as in Hobbs and Shieber's algorithm represent predicate argument relations but not quantifier scoping information. The input formulae will also not indicate which pronouns are anaphorically related to which antecedents but it will indicate whether pronouns are reflexive or not. The implicit claim in this is that an interesting amount of work can be done using just these syntactic properties. Obviously, other constraints, such as gender, could be added too. In the implementation (see Appendix A.3), the input data structures are produced whilst parsing a Definite

Clause Grammar using the standard Prolog top-down left-to-right control strategy.

The definition of the input formulae is as follows (where *ref* and *nonref* stand for *reflexive* and *non-reflexive* respectively)

RL+ **Formulae**

1. If x is a variable and α is an individual constant then $\mid x\ \alpha \mid$ is proper name term
2. If $\alpha \in \{ref, nonref\}$ and x is a variable, then $\langle x\ \alpha \rangle$ is a pronoun term
3. If ϕ is a formula, and x is a variable, then $< \exists\ x\ \phi >, < \forall\ x\ \phi >$ are complex terms.
4. If $t_1, \ldots, t_n$ are complex terms, proper name terms, pronoun terms or formulae, and P is an n-ary predicate, then $P(t_1, \ldots, t_n)$ is a formula.

The assumptions behind the input formulae are that each noun phrase in a discourse will give rise to a term containing its own (unique) variable. In particular, the variable in each pronoun term $\langle x\ \alpha \rangle$ will be unique to that term. The same variable may occur more than once in an input formula only if the input syntactic structure indicates that an argument is an argument to more than one predicate. In the Rooth fragment this only arises for prepositional phrases. For example, 60a is represented by 60b in the input language

(60) a Every woman with a son loves him
b $loves(< \forall\ x\ and(woman(x), with(x, < \exists, y, son(y) >)) >, \langle z, nonref \rangle)$

Here x occurs both as an argument to *woman* and also to *with*.

The output data structures are defined as follows

1. There is a basis set containing individual constants, variables and n-ary predicates
2. If $t_1, \ldots, t_n$ are individual constants or variables and P is an n-ary predicate, then $P(t_1, \ldots, t_n)$ is a formula.
3. If ϕ is a formula, and X is a sequence of variables, then $\exists(X, \phi)$,$\forall(X, \phi)$ are formulae.

For example, the input formula

$loves(< \forall\ x\ and(man(x), with(x, < \exists, y, daughter(y) >)) >, \langle z, nonref \rangle)$

will be transformed into

$\forall([x], \forall([y], -> (and(man(x), and(daughter(y), with(x, y))), loves(x, y))))$

Before describing how one transforms input formulae into output formulae, let us briefly review some lessons from chapter 3 on algorithms for quantifier scoping.

First, one inspiration for such algorithms is the hope that one can first (compositionally) produce a meaning representation which leaves quantifier scoping issues unresolved and then run a separate specialised quantifier scoping module. The hope is fortified by the observation that the compositional production of an unscoped meaning representation is simpler than that of a scoped one since the issue of quantifier scoping itself raises awkward theoretical issues about the syntax-semantics interface. Furthermore, since giving a quantifier one scope rather than another can be viewed as just inserting a quantifier symbol into one place rather than another in a structure, the specialist module will not have to reassess drastically the overall structure of the meaning representation.

The difficulty we encountered with such an algorithm was explaining why it was successful. How could we relate it to a semantic theory? Part of the answer, I suggested, was simply to be clear about object and meta-languages. In particular, the outputs of the algorithm at any stage were to be viewed as part of the meta-language whereas the inputs were object language syntax. The algorithm was not therefore modifying formulae of one language but, like a truth theory, giving the properties of one language in another. The reason why it was so easy to view the algorithm as continually altering a representation - and indeed why the algorithm is actually so simple - is simply that the object and meta-languages are so close to each other. Again, the meta-language differs from the object language virtually only in where the quantifier symbols appear.

At first sight it may appear that at least some of these lessons will not apply in our

dynamic environment. In particular, we cannot view scoping just as the question of where to place a quantifier symbol in an existing structure - since, for example, there are subtle interactions in dynamic logic between quantification and discourse context. Also, the object and meta-languages, for DPL and RL+ at least, are not as close as they were for HSL. The meta-languages we have used for DPL and RL+ have been statically interpreted whereas the object languages are dynamic. Furthermore, in dynamic interpretation we have to maintain two pieces of information - the current context and the truth conditional content, whereas in our earlier algorithm we were only interested in generating truth conditional content.

The fact that the object and meta-languages are not very close could be overcome by re-formulating the truth theory for a dynamic logic *in* a dynamic logic. However, the point of producing semantic representations is precisely that they should be of some further use in, for example, reasoning. In this respect, it is more sensible not to generate meaning representations in a language such as dynamic predicate logic, whose proof theory remains relatively undeveloped (for some recent work on proof theory in a variant of DPL see [van Eijck & de Vries 91]). Even if all we use the representations for is checking that the correct predictions are made for the input sentences, it is probably better to produce representations in a familiar language such as FOPL.

The difference between object and meta-languages in our algorithm means that some more complex meta-language inferences must be encoded. In our earlier algorithm, most of the inferences we employed were simply of the form 'P if Q, Q if R therefore P if R'. One example of a meta-language inference that we will now make is

From:
$\exists k\ \exists \alpha\ (\emptyset_x^\alpha\ =\ k\ \wedge\ [\![\ x\ ]\!]^{\emptyset_x^\alpha}\ \in F(m))\ \wedge\ [\![\ x\ ]\!]^k\ \in F(w)$
Infer:
$\exists \alpha\ (\alpha\ \in F(m))\ \wedge\ \alpha\ \in F(w)$

The former formula is what one would might write down as a result of simply working through the truth theory - however, one only wants to generate the simpler, equivalent latter formula as the final meaning representation. In particular, the initial $\exists k$ is likely

to derive from initial use of the rule that a formula is true just if *there is* an output context such that the formula is true with respect to $\emptyset$ and that output context. The occurrence of $\exists\alpha$ will derive from analyzing an object language existential quantifier. Following the point we made earlier in chapter two concerning whether DPL really rejects the 'indefinites as variables thesis', the algorithm treats object language existentials as simply introducing free variables. This is justified by the fact that they will always occur in the form $\exists\alpha$ ($\emptyset^{\alpha}_{x} = k$ ... where k is governed by a higher quantifier, and therefore $\exists\alpha$ will not contribute any existential force to the resulting formula in any case, as we noted earlier. A recursive formula may, of course, introduce several such 'free variables' and an outer quantifier over contexts (such as $\exists k$ in the above example) may therefore quantify over several such variables at once. The algorithm therefore calculates which variables should be quantified at which stage - but this does not mean reneging on our commitment in chapter 3 not to appeal to the syntax of the meta-language, because in each case the step can be justified by a valid meta-language inference like the simple one stated above.

As a result of evaluating a formula in a dynamic logic, one derives two results rather than one. First, there is the truth conditional content and, secondly, there is the resulting output context. In our lisp-like language we will therefore employ some new language constructs to deal with two-valued functions. The construct **two_values** takes two arguments and returns the two evaluated arguments. There is also a new form of assignment

```
let X,Y := exp
in body
```

where `X` and `Y` are variable names, `exp` is an expression that returns two values and `X` and `Y` are assigned respectively to the two results of evaluating `exp`.

We will represent an RL+ context as a list of lists, each list of which will represent a sequence in reverse order. So the *last* member of the list will be the object which serves as the value of all OTHER members of the list. In our earlier algorithms both

constants and variables evaluated to themselves (though we could distinguish object and meta-language uses by underlining) We shall do the same here by making the last member of a list (the object denoted by all other members of the list) the variable of the second last member of the list (or the individual constant α if the last member derives from the representation of a proper name, $| x\ \alpha |$). As an example, the following sentence

[A man]$_1$ with John$_2$ beat him$_3$.He$_4$ despised himself$_5$

will result in either of two possible representations of contexts: $\langle\langle 5,4,1,1\rangle\langle 3,2,\text{John}\rangle\rangle$ and $\langle\langle 1,1\rangle\langle 5,4,3,2,\text{John}\rangle\rangle$. The initial context will be $\langle\rangle$.

(In the following discussion, I shall ignore proper names. However, in the implementation, proper names are treated simply as discourse wide quantifiers. The first step of the algorithm is then to expand the initial context of $\langle\rangle$ by adding in $\langle x\ pn\rangle$ for each proper name term $|\ x\ pn\ |$ in the discourse. In the above example, $\langle\rangle$ would be expanded to $\langle\langle 1, \texttt{john}\rangle\rangle$. Since each proper name begins a new sequence within a context, this maintains Higginbotham's principle **C'** that a full referring expression is not to be linked.)

Algorithmic detail

The first call in the algorithm *pull/2* introduces the effect of existential quantification over an output context because a discourse is true if there is an output context such that it is true with respect to $\langle\rangle$ and that output. Quantifying over a context essentially 'mops up' any variables which, were they to occur in a subsequent discourse, *would* be bound by a quantifier in the existing discourse. This step is implemented in the algorithm by the function *simplify* which simply checks what is 'available' for subsequent discourses and existentially quantifies over them.

```
function pull(form,inputcontext);
  let lf,output := pull_two(form,inputcontext)
  in  two_values(simplify(inputcontext,output,lf),output)
```

simplify is defined as follows where *difference* returns a list of objects in the output which were not in the input. *ewff* returns a formula of form $\exists(diff, lf)$ when supplied with $diff$ and lf as arguments.

```
function simplify(inputcontext,output,lf);
  let diff := difference(inputcontext,output)
  in  if  null(diff)
      then lf
      else ewff(diff,lf)
```

For example, if the inputs were $ran(\langle \exists\ x\ man(x) \rangle)$ for the truth conditional content and $\langle\rangle$ for the context, and the results from *pull_two/2* were $and(man(x), ran(x))$ for lf and $\langle\langle x \rangle\rangle$ for *output*, then *diff* would be calculated to be x and so *ewff/2* (and hence *simplify/3* and *pull/3* itself) would be $\exists([x], and(man(x), ran(x)))$.

The main recursive procedure *pull_two/2* is, as usual, case based depending on a top-down analysis of the syntactic structure, where *terms/1*,*conj/1* and *imp/1* return true just in case the input form contains an applicable complex term, a conjunct or an implication. An applicable complex term will be one not nested inside a conjunction or implication.

```
function pull_two(form,inputcontext);
  if    terms(form)
  then apply_terms(form,inputcontext)
  else
       if    conj(form)
       then apply_and(form,inputcontext)
       else
            if     imp(form)
            then apply_imp(form,inputcontext)
            else  apply_atomic(form,inputcontext)
```

The function for conjunction is the most straightforward. *cwff/2* is a function that takes two arguments a and b and returns and([a,b]). We simply evaluate the first conjunct in the input context and thread the resulting output context into the evaluation of the

second conjunct. The result is the conjunction of the results of the two conjuncts. The output context for the function is the output context for the second conjunct.

```
function apply_and(form,inputcontext);
  let and(c1,c2) := form
      LF_1, Inter := pull_two(c1,input)
      LF_2, Output := pull_two(c1,Inter)
  in two_values(cwff(LF_1,LF_2),Output)
```

The function for applying a complex term is very similar to that employed in earlier chapters. The function differentiates between existential and universal quantifiers both in the output context returned and in the recursive call. In particular, as noted above analysis of an existential quantifier in the input language no longer generates an existential quantifier in the output language but only a 'free' variable. *add_to_context/2* is a function that takes a context and an index v and returns a new context like the old one but containing in addition $\langle v, v \rangle$. *awff/3* takes a variable v and a logical form lf and returns $\forall(v, lf)$.

```
function apply_terms(form,inputcontext);
  let < q v r > := applicable_term(form)
        body    := subst(v,< q v r >,form)
      newcontext := add_to_context(inputcontext,v))
  in  if   q = ∃
      then pull_two(cwff(r,body),newcontext)
      else two_values (awff(v,pull(iwff(r,body),newcontext)),inputcontext)
```

For example, if the input is $runs(< \exists\ x\ man(x) >)$, then we simply end up recursing on $and(man(x), runs(x))$ with an input context containing a new sequence $\langle x\ x \rangle$. The result will be just be $and(man(x), runs(x))$ with the meta-language quantifier over x only being generated by a higher call to *simplify/3* - that call will note the difference between the relevant input context and output context which will include the x just added in by *add_to_context/2*.

The function for implication resembles that for conjunction. As with the main call *pull/2*, an implication introduces a quantifier over a whole context. The result is

quantification over anything existentially quantified over in the antecedent of the conditional. *simplify_imp/4* is a step corresponding to the following sort of inference in the truth theory (where *iwff/2* returns $\rightarrow (a, b)$ where a and b are its two arguments)

From:
$\forall k\ (\exists \alpha\ (\emptyset_x^\alpha\ =\ k\ \wedge\ [\![\ x\ ]\!]^{\emptyset_x^\alpha}\ \in F(m))\ \rightarrow\ [\![\ x\ ]\!]^k\ \in F(w)\ )$
Infer
$\forall \alpha\ (\alpha\ \in F(m)\ \rightarrow\ \alpha\ \in F(w))$

and is defined as follows

```
function simplify_imp(inputcontext,output,lf_1,lf_2);
  let diff := difference(inputcontext,output)
  in  if  null(diff)
      then iwff(lf_1,lf_2)
      else awff(diff,iwff(lf_1,lf_2))
```

For example, an input of $if(owns(< \exists\ x\ man(x) >, < \exists\ y\ donkey(y) >), feeds(u, v))$ will result in the following output for the antecedent

$and(man(x), and(donkey(y), owns(x, y)))$

plus an output context which contains both $\langle x\ x \rangle$ and $\langle y\ y \rangle$ over and above whatever the input context is. *simplify_imp/4* will therefore generate a universal quantification over the pair $[x, y]$. (How the formula $feeds(u, v)$ is dealt with is explained below).

```
function apply_imp(form,inputcontext);
  let  —(c1,c2)  := form
       LF_1, inter := pull_two(c1,inputcontext)
       LF_2,       := pull(c2,Inter)
  in two_values(simplify_imp(inputcontext,inter,LF_1,LF_2),inputcontext)
```

The base case is for atomic formulae where we alter the input context by adding in the indices for all reflexive and non-reflexive pronouns in accordance with the restrictions stated in the previous chapter, and then return the same formula with all indices replaced by the values according to the new context.

5.2.1 Examples

I shall informally describe two examples. The outputs of a large number of examples can be found in Appendix A.3.4.

First let us work through the algorithm with the representation of 'a man owns a donkey':

$owns(< \exists\ x\ man(x) >, < \exists\ y\ donkey(y) >)$

The initial input context is $\langle\rangle$ and there are no proper name terms so the input context for *pull* will be unchanged - $\langle\rangle$. Next, *pull_two* will be called with the same inputs. There are two applicable complex terms so *apply_terms* will be called next. We will suppose $< \exists\ x\ man(x) >$ is the first to be returned by *applicable_term*. The local variable `body` will be assigned to $owns(x, < \exists\ y\ donkey(y) >)$. Then, we will recurse on *pull_two* with the formula $and(man(x), owns(x, < \exists\ y\ donkey(y) >)$ and an input context of $\langle\langle x\ x\rangle\rangle$. *apply_and* will be called next and its output will be the result of recursions on $man(x)$ and $owns(x, < \exists\ y\ donkey(y) >)$. The former is an atomic case and so the result will be the same but with all indices replaced by their values. According to the context $\langle\langle x, x\rangle\rangle$ the value of x is x, therefore the result is $man(x)$. The second conjunct again contains a complex term and its output will be the result of recursion on $and(donkey(y), owns(x, y))$ - which will simply be $and(donkey(y), owns(x, y))$, plus an output context of $\langle\langle x, x\rangle\langle y, y\rangle\rangle$. Therefore, the result of the recursion beginning at the initial call to *pull_two* will be $and(man(x), and(donkey(y), owns(x, y)))$. Notice that despite having analysed two singular indefinites no existential quantifiers have yet appeared in the output. The results of the initial call to *pull_two* including the input context $\langle\rangle$ and $\langle\langle x, x\rangle\langle y, y\rangle\rangle$ are passed to *simplify* which calculates the difference between them as x, y. The resulting formula is therefore given as $\exists([x, y], and(man(x), and(donkey(y), owns(x, y))))$.

Secondly, let us consider 'If a man owns a donkey, he beats it' as represented by:

$\rightarrow (owns(< \exists\ x\ man(x) >< \exists\ y\ donkey(y) >), beats(\langle z\ nonref\rangle, \langle q\ nonref\rangle))$

Once again, the input to the initial call to *pull_two* will be $\langle\rangle$. Since the major connective is an implication *apply_imp* will be called next. The first sub-call of *pull_two* with $owns(< \exists\ x\ man(x) >< \exists\ y\ donkey(y) >)$ is identical to our previous example. The result is that the local variable LF_1 will be $and(man(x), and(donkey(y), owns(x, y)))$ and `inter` will be $\langle\langle x, x\rangle\langle y, y\rangle\rangle$. The second call is to *pull* with the arguments being $beats(\langle z\ nonref\rangle, \langle q\ nonref\rangle))$ and $\langle\langle x, x\rangle\langle y, y\rangle\rangle$. This is an atomic case. We can add the indices z and q into either sequence in the context but not both into the same one. Here we will let the context be altered to $\langle\langle z, x, x\rangle\langle q, y, y\rangle\rangle$. Then, we replace the variables with their values generating $beats(x, y)$ as the value of `LF_2`. The final call in *apply_imp* is to calculate the difference between its input $\langle\rangle$ and the output $\langle\langle z, x, x\rangle\langle q, y, y\rangle\rangle$ and universally quantify over it. The difference is x, y and so the resulting formula from *apply_imp* is

$\forall([x, y], \rightarrow (and(man(x), and(donkey(y), owns(x, y))), beats(x, y)))$

The resulting context from *apply_imp* is simply $\langle\rangle$ which is the same as its input. Therefore, the final call to *simplify* on exiting *pull_two* leaves the formula unchanged.

5.3 Comparisons with other Systems

5.3.1 Latecki and Pinkal's Suggestion

Latecki and Pinkal have suggested a somewhat similar treatment of 'donkey anaphora' and quantifier scope ambiguities in their [Latecki & Pinkal 90]. In particular, they are concerned with 'weak crossover' cases such as 'A friend of his saw every man'. The reason is that if one quantifies in 'every man' then the pronoun 'his' falls within the semantic scope of 'every man'. Yet 'his' still cannot be bound by 'every man'. That is, the sentence cannot be read as claiming: for each man, a friend of that man saw that man. They propose to use syntactic information to rule out such a case. Their rule is

that a pronoun can be bound when quantifying in only if the noun phrase at which the quantifier was constructed c-commands the pronoun in question. Since 'every man' does not c-command 'his', it follows that 'his' is not allowed to be bound by 'every man' in 'A friend of his saw every man'. Since Latecki and Pinkal work in a DRT framework where singular indefinites are not quantifying phrases, their rule allows 'a donkey' to be bind 'it' in

Every man who owns a donkey beats it

The reason is that the phrase we actually quantify in is 'every man who owns a donkey', which, in the DRT framework, amounts to quantifying over pairs of men and donkeys. Since this phrase *does* c-command 'it', 'it' can be bound whilst quantifying in and be equated to the second member of the pair. Latecki and Pinkal do not state additional restrictions to prevent 'it' being bound by 'every man', although this should not prove a serious difficulty. They also do not consider inter-sentential cases of anaphora.

In contrast, our account allows 'a donkey' to bind 'it' not because of any c-command relations but because the two are not *local* and 'a donkey' does precede 'it'. As mentioned in the previous chapter, typical cases where locality proves superior to c-command are PP cases such as 'I talked to every student about his examination mark.' Similar donkey cases are rather harder to construct, though perhaps not impossible. Witness

(61) a I talked to every student who failed an examination about it
b I talked to every mother who had a son at school about him

The two accounts can also differ on cases such as

(62) a man who knows every woman loved her

assuming one allows 'every woman' to take scope outside the relative clause. In this case, 'every woman' and 'her' are non-local and therefore our account would predict a possible binding. On Latecki and Pinkal's account, however, 'every woman' does not c-command 'her' and therefore a binding is not possible. The correct account of the

matter, I suggest, is that 'every woman' should *not* take scope outside of the relative clause and so 62 is not a counterexample.

Even if one insists that there are clear cases of nested noun phrases where the inner noun phrase takes wide scope, it is not entirely clear that the c-command rule is correct. Latecki and Pinkal cite

(63) A friend of every man saw him

in their favour where 'every man' can outscope 'a friend' and yet the reading where 'every man' binds 'him' seems unnatural. However, it is not difficult to find similar examples which are more convincing.

(64) a Somebody from every city despises its architecture (from [May 88])
b A friend or close relative of each suspect confirmed his alibi

May also claims that in 64a 'every city' clearly does bind 'its'. His reason is that there is a strict/sloppy ambiguity in

(65) Nobody from New York rides its subways but everybody from Tokyo does

That is, the reading that everybody from Tokyo rides Tokyo's subways demands an analysis where 'its' is bound by the embedded complement position.

5.3.2 Pereira and Pollack's System: Candide

A second system which incorporates 'donkey anaphora' is Pereira and Pollack's system 'Candide' [Pereira & Pollack 91]. The general aim of Candide is to 'account for the influence of context on interpretation while preserving compositionality to the extent possible'. To this end, phrases are assigned *conditional interpretations* of form $A:s$ where s is the 'sense' of the phrase and A is a set of assumptions which constrains how the sense may be connected to the context. For example, 'the donkey' may be assigned $bind(x,def,donkey):x$ meaning that the sense of 'the donkey' is x where x is

bound to be of the sort *donkey* and will fall under the rules for definite reference. A conditional interpretation (*ci*) is derived both by structural rules, which give the *ci* for the whole in terms of those for its parts, and also by discharge rules, which remove assumptions from the set A and update the sense s. For example, given the above *ci* for 'the donkey', one could discharge the 'bind' assumption by finding a unique donkey in the discourse model d and derive a new *ci* []*:d*, where [] represents the empty list of assumptions.

The structural rule for a quantified noun phrase builds a *ci* containing a special *bind* assumption which can be discharged 'higher' up the structure tree. For example, 'every donkey' would be assigned the *ci bind(x,every,donkey):x* meaning that x is of type donkey and is to be universally quantified over. By placing this information in an assumption list which can be passed up the structure tree, the effect is rather like putting a quantifier in a Cooper store. Assumptions can also be nested thereby allowing a version of nested Cooper Storage. Discharging a quantified bind assumption is equivalent to quantifying in. The discharge rule is (leaving aside nested assumptions)

if node P has *ci A,bind(x,q,s):p*
(where A is a set of assumptions, q is a quantifier
p is of type formula and x is not free in A)
then node P has *ci A,B:(q,s,x)p*

The *ci* for 'Most spectators cheered every competitor' would be

[bind(s,most,spectator),bind(c,every,competitor)]:cheer(s,c)

By discharging the assumptions in either order one derives the two readings of the sentence.

(most spectator s)(every competitor c)cheer(s,c)

(every competitor c)(most spectator s)cheer(s,c)

Interestingly, and for reasons not made entirely explicit, singular indefinites are not treated as quantifiers. The structural rule for a node such as 'a donkey' introduces a 'referential' expression *bind(x,indef,donkey):x*. The reason for this appears to be to allow certain inter-sentential anaphora. There are two ways to resolve a pronoun. Either one can look to the discourse model for recently mentioned objects or one can look for parameters present in the assumption list of the current node. Assumptions are not carried between sentences so the only way to secure intersentential anaphora is to assume that the indefinite antecedent introduced an object into the discourse. Normally, the treatment of singular indefinites as referring expressions would prevent a scope ambiguity being detected in 'every man owns a donkey' therefore Candide also allows the quantification discharge rule to apply to indefinite bind assumptions under certain special circumstances. It follows that 'a donkey' is treated ambiguously. However, such a treatment still does not extend to relative clause donkey sentences where a third interpretation mechanism is required. A special rule called 'capture' is triggered by the presence of indefinite bind assumptions nested within a quantified noun phrase bind assumption. For example, the *ci* corresponding to 'every driver that controls a jet' will contain a bind assumption for 'a jet' within the list of its nested assumptions. The rule can discharge such a nested indefinite bind assumption and generate an appropriate polyadic quantified output.

From the perspective of dynamic logic, Candide's analysis is an interesting one. For even though a discourse context was used to handle intersentential anaphora, the procedure does not work for the intra-sentential cases. The reason is that the domain only contains particular objects mentioned and in the phrase 'every farmer who owns a donkey' there is no particular donkey referred to or mentioned at all. Furthermore, 'a donkey' cannot be treated as a quantifier and bind a pronoun in a verb-phrase such as 'beats it' for this would violate Pereira's version of the 'free variable constraint' namely the condition in the discharge rule that x not appear free in A. This is not surprising since that constraint is justified by what derivations are allowable in an ordinary

non-dynamic functional calculus ([Pereira 90]). A similar calculus allowing dynamic combinations, if such could be devised, would no doubt justify quite different inferences. In order to justify the sort of interpretation rule used for the donkey anaphora, one needs a quite different background semantics - for example, that provided by the small fragment of the previous chapter.

5.3.3 The SRI Core Language Engine

The CLE ([Alshawi *et al* 88],[Alshawi 90],[Alshawi 92]) is designed to embody an 'extreme form of a staged natural language processing architecture whereby modular processing components have the function of mapping between different levels of sentence representation'. In particular, morphological, syntactic and semantic rules produce a *Quasi-Logical Form* (QLF), which like the input to our algorithm does not resolve quantifier scope ambiguities or anaphora. The CLE handles a wide range of grammatical constructions and also incorporates other 'resolutions' such as distributive versus collective readings and referential versus attributive readings. From QLFs are generated RQLFS (resolved QLFs) by quantifier scoping and pronoun resolution rules. RQLFs are then filtered through various linguistic and non-linguistic plausibility constraints. Here I shall restrict myself to the way the CLE handles quantifier scope ambiguities and pronoun resolution.

The CLE invokes a quantifier scoping algorithm similar to that of Hobbs and Shieber, which progressively modifies the input QLF by 'raising' embedded complex terms subject to various conditions. In particular, Pereira's 'free variable constraint' is explicitly coded as a constraint on possible logical forms. In forthcoming work however [Alshawi & Crouch 92], a semantics for QLF is given which, like our account of chapter 3 and [Lewin 90], makes essential use of conditionals and not biconditionals in the theory and allows the 'free variable constraint' to emerge quite naturally from the way the quantification rules are applied. No filter on possible logical forms needs to be applied. There is no destructive modification of input structures.

The CLE also has a small dynamic aspect to its processing. A discourse model is maintained containing entities referred to plus some salience weights which allows recently mentioned items to be preferred. New entities are introduced into the model only after a whole sentence has been processed. The account again allows inter-sentential cases of anaphora (bar 'one anaphora') only when the antecedents have genuinely referred. Since the CLE does not follow Candide in allowing singular indefinites a referring role, some typical cases of intersentential dynamic anaphora are not allowed. The CLE does have a special mechanism for relative clause donkey anaphora. In general in the CLE, pronouns may not be resolved to quantified noun phrases if they do not occur in the quantifier's semantic scope (this is the 'free variable constraint' again). However, variables from an indefinite noun phrase in the restriction of a quantified noun phrase are treated as a special case. In this case, the pronoun can be replaced by the same variable as that of the indefinite noun phrase - however, a marker which normally indicates that processing is not complete is left in place, indicating that the structure is special. Unlike Candide, no attempt is made to rewrite the formula into an ordinary non-dynamic language.

5.4 Conclusions

I have presented an algorithm for generating representations in a non-dynamic logic for input structures which are not resolved with respect to quantifier scope ambiguities and anaphora. The algorithm uses dynamic logic to generate its results. I have also analysed three other current approaches to donkey anaphora in the current literature. Latecki and Pinkal's approach is the more linguistically oriented of the three in spirit and uses c-command relations rather than locality to produce their results. They do not discuss inter-sentential donkey cases. The latter two approaches both have a much wider coverage of linguistic data than the small algorithm presented here. However, they do both attempt to cover some discourse data too and both employ discourse contexts which are updated during processing. Neither is entirely successful

in their attempts and the conclusion is easily reached that the fundamental assumption behind both approaches is that contexts contain objects that have been referred to and there is no quantification over possible contexts. The special rules to handle some discourse cases are adopted on a rather ad-hoc basis. Indeed, the background justification for the quantifier rule in Candide will not extend in any straightforward way to the special discourse anaphora rules. If one wants to accommodate the discourse data in a principled fashion, then I suggest an approach along the lines of the algorithm of this chapter is to be recommended.

Chapter 6

Dynamic Binary Quantificational Structures

6.1 Introduction [1]

Two of the most significant developments in formal semantics over the last decade have been those of generalized quantifier theory ([Barwise & Cooper 82], [van Bentham 86]) and dynamic semantics. The union of two such attractive individuals naturally appears to be a most promising partnership. Nevertheless, the marriage of the two has proved rather difficult to forge without sacrificing some of the interest which separately attaches to the betrothed. I have already noted in chapter 4 three difficulties with current attempts to incorporate generalized quantifiers into dynamic semantics. The first concerned how dynamic the approach actually was given that it involved the evaluation of material in more than one place in order to make the right information available at different points in the analysis. The second difficulty concerned the thorny empirical issue of what 'donkey' sentences do in fact mean. The third difficulty concerned how the strategy could be reconciled with our attempt to give a dynamic semantics for a natural language where the issue of pronominal binding is not settled in advance purely by syntactical considerations.

In this chapter, I explore the first two issues in greater depth and propose a new method

[1] Part of the content of this chapter has appeared in [Lewin 91]

for incorporating quantifiers that take two arguments into dynamic logic. This method, unlike other current attempts, does thread information from the first argument of a binary quantifier into the second argument and does not make the information available simply by re-evaluating the relevant material again in a suitable position. The resulting system has some other interesting properties including a separation of the rules for context change from those giving truth conditional content (the two are not independent, of course, but they are distinct) and the fact that the truth conditional rules are 'homophonic' in that each object language operator is analysed by a corresponding operator in the meta-language (object language conjunction by meta-language conjunction and so on). I shall not, however, here address our third concern of giving a similar semantics for a suitably ambiguous language. I do not believe the union of a system with dynamic binary structured quantifiers together with the sort of analysis I gave for the Rooth fragment in chapter 4 to be impossible; but it may prove to be of formidable complexity and I shall not attempt the combination in this thesis.

The rest of this chapter is structured as follows. First, I briefly recall the treatment of quantifiers both in DPL and in Generalized Quantifier Theory. Then, I present an argument to show that generalized quantifiers cannot be incorporated into a dynamic logic which threads information between the two arguments if contexts are taken to be single assignments, as they are in DPL. Secondly, I discuss two recent attempts in DPL and DRT to include generalized quantifiers in a dynamic environment and show that both attempts make use of extra operations beyond that of their peculiar dynamic mechanisms. As a result of this analysis, I conclude by presenting an alternative system $\mathrm{DPL}_{\mathrm{bq}}$ which meets our requirements by handling binary quantifiers and anaphora without making use of a double evaluation and which does thread the output context for the first argument into the input of the second argument.

6.2 DPL and Binary Quantification

The quantifiers defined for DPL (2.2) are unary operators. They take one argument only. In order to translate English quantified sentences that include a restriction, such as 'man' in the sentences 'A man runs' and 'every man runs', Groenendijk and Stokhof make use of Frege's trick of the hidden connective. That is, 'A man runs' is true just in case there is something which has the complex property of 'being a man and running'. Also, 'every man runs' is true just in case everything has the complex property that 'if it is a man then it runs'. Formally, we translate the two sentences as

$\exists x\ (man(x)\ \wedge\ runs(x))$
$\forall x\ (man(x)\ \rightarrow\ runs(x))$

Incidentally, since the conditional $\rightarrow$ possesses universal force in DPL, this analysis predicts that a sentence such as 'Every man who owns a donkey feeds it' is true just in case every man feeds every donkey that he owns. This prediction is the subject of some dispute nowadays (the favoured counterexample being 'Every man who has a dime puts it in the meter', which is not taken to imply that every man puts all of his dimes into the meter). It is worth noting that the prediction from DPL arises solely from the universal force found in the hidden connective used in the analysis of 'every'.

For reasons familiar from the literature on Generalized Quantifiers, we cannot handle generalized quantifiers using the technique of hidden connectives or indeed in FOPL (*cf.* [Rescher 62], [Barwise & Cooper 82]). Generalized quantifiers require a binary structure where the value of a determiner such as 'most' takes two sets of objects as arguments, and yields a truth value. One way to state this is to say that a determiner combines with two open formulae to yield a formula, thus

> If ϕ,ψ are formulae, and x is a variable, then $\exists x\ (\phi;\psi)$, $\forall x\ (\phi;\psi)$, and $Mx\ (\phi;\psi)$ are formulae.

In these formulae, ';' functions purely as a delimiter between the two arguments and should not be confused with the same symbol as used in [Groenendijk & Stokhof 91a],

for example where $(\phi; \psi)$ is a well-formed formula denoting the dynamic conjunction of ϕ and ψ.

The semantic rules for these formulae in a non-dynamic environment may be given quite straight-forwardly in an informal meta-language, English, as follows

$[\![\exists x(\phi; \psi)]\!]^g = 1$ iff some α such that $[\![\phi]\!]^{g^\alpha_x} = 1$ is also such that $[\![\psi]\!]^{g^\alpha_x} = 1$
$[\![\forall x(\phi; \psi)]\!]^g = 1$ iff every α such that $[\![\phi]\!]^{g^\alpha_x} = 1$ is also such that $[\![\psi]\!]^{g^\alpha_x} = 1$
$[\![Mx(\phi; \psi)]\!]^g = 1$ iff most α such that $[\![\phi]\!]^{g^\alpha_x} = 1$ are also such that $[\![\psi]\!]^{g^\alpha_x} = 1$

I shall abbreviate these rather long-winded meta-language statements by instances of the following schema

$$[\![Qx\ (\phi; \psi)]\!]^g = 1 \text{ iff } Q'\alpha\ ([\![\phi]\!]^{g^\alpha_x} = 1\ ;\ [\![\psi]\!]^{g^\alpha_x} = 1)$$

The notation in the meta-language is thereby made to resemble that in the object language. The analysis of an object language quantifier is given by the *use* of a corresponding meta-language quantifier whose domain of quantification is the set of objects in the model. Furthermore, closed and open formulae (of arbitrary complexity) are all treated alike as satisfiable by partial assignments. Any occurrence of a particular quantifier is always treated by the same rule as all other occurrences. All the quantifiers are treated in the same fashion.

It should be fairly clear that this method of exposition is equivalent to one which defines generalized quantifiers by means of the cardinality of the two sets denoted by A and B in $Q\ (A; B)$ ([van Bentham 86] contains a discussion of when this is possible). If you object to the informal meta-language, then please take each instance of the schema and understand each Q' according to your favourite formal version of generalized quantifier theory.

Within ordinary logic, the move from unary to binary quantifiers seems to be a fairly straightforward one. Can we make the same move as easily in a dynamic environment ? Naturally, we will want to be able to thread information between the two parts of the binary structure. That is a requirement in virtue of the possibility of 'donkey anaphora' in sentences such as 'Most farmers who own a donkey feed it'. We will

also want to have a genuinely binary quantificational structure - an object language structure whose analysis is given by a binary structure in the meta-language, since the 'hidden connective' technique does not give us the expressive power of generalized quantifiers. It is meeting these two aims which is the goal of this chapter.

6.2.1 Threading single assignments

In the definition of DPL (2.2), the key semantic notion characterized recursively is 'truth of a formula ϕ with respect to a pair of assignments'. If $\phi \otimes \psi$ is any complex formula with binary operator $\otimes$ and whose parts are ϕ and ψ, then the definition shows how the truth of $\phi \otimes \psi$ (relative to some pair of assignments) depends on the truth of ϕ (relative to a pair of assignments) and the truth of ψ (relative to some pair of assignments). Similarly, if Γ is any quantifier, then the truth of $\Gamma x\ (\phi)$ (with respect to a pair of assignments) is shown to depend on the truth of ϕ (with respect to a pair of assignments). In this recursive way, the demand for compositionality is met.

If we are to extend DPL to include binary structured quantifiers then we shall want to show how the truth of $\Gamma x\ (A; B)$ (with respect to a pair of assignments) depends on the truth of A (with respect to a pair of assignments) and the truth of B (with respect to a pair of assignments). Keeping to the standard explication of quantification, we will find ourselves writing down something along the lines of the following schema. In the schema W,X,Y,Z are to be filled in later and the meta-language quantifier is distinguished from its object language counterpart by underlining.

Binary Quantifier Schema
$[\![\ \Gamma x\ (A; B)\ ]\!]^{i\ o}$ iff $\underline{\Gamma}\alpha\ ([\![\ A\ ]\!]^{W\ X}\ ;\ [\![\ B\ ]\!]^{Y\ Z})$

If we can just fill in the values of W,X,Y and Z then we will have once more analyzed an object language quantifier by using a corresponding meta-language quantifier over objects in the domain. We will also have given a recursive account of the satisfaction of the whole in terms of that of the parts which will allow for arbitrary nesting of object language quantifiers.

We can fill in W immediately as i_x^α. What can X be ? X must be allowed to vary with W because our base clause (namely, $[\![P(t_1,\ldots,t_n)]\!]^{i\ o}$ iff $i = o \wedge < [\![t_1]\!]^i,\ldots,[\![t_n]\!]^i > \in F(P)$) states that W and X are identical for atomic ϕ. So, if W varies so must X.

One possibility for letting X vary with W is to quantify over both by $\underline{\Gamma}$ in the meta-language. That is, $\underline{\Gamma}$ will quantify over pairs of individuals and Output contexts. Our schema will be partially completed in the following way:

Modified Schema
$[\![\Gamma x\ (A;B)]\!]^{i\ o}$ iff $\underline{\Gamma} < \alpha, g >\ ([\![A]\!]^{i_x^\alpha\ g}\ ;\ [\![B]\!]^{Y\ Z})$

Then, in order to invoke the notion of threading between the two arguments we can just instantiate Y to g. However, as is well known, this type of solution is bound to lead us into the *Proportion Problem* (*cf.* [Partee 83]). Suppose we represent 66a by 66b

(66) a Most men that own a donkey are fed-up
b *Most* $x(man(x) \wedge \exists y\ (donkey(y); own(x,y)); fedup(x))$

Suppose 66a is the first sentence of the discourse so that $i = \emptyset$. Then our Modified Schema for evaluating 66b tells us to find the pairs of individuals α and output assignments g for which $\emptyset_x^\alpha$ and g satisfy 'man that owns a donkey'. Now suppose some man 'j' owns two donkeys: 'p' and 'q'. Then the pair $< \emptyset_x^j, \emptyset_{xy}^{jp} >$ satisfies 'man that owns a donkey, *and so does* $< \emptyset_x^j, \emptyset_{xy}^{jq} >$. The Modified Schema will therefore count men-donkey pairs that stand in the ownership relation, and not just the men who are donkey-owners. So even if there is only one other man who owns a donkey and he owns exactly one donkey and he is not fed-up, our Modified Schema will count 66a true. The reason is that there are two men-donkey pairs (where the man owns the donkey) where the man *is* fed-up and only one where the man is *not* fed-up. Since the former outnumber the latter the sentence is deemed to be true. But, in this circumstance, only half of all men who are donkey-owners are actually fed-up and that is not sufficient to warrant the truth of 66a. This, briefly, is the Proportion Problem.

We have been considering the possibility of quantifying over the output context for A

with $\underline{\Gamma}$. Another possibility which permits X to vary with W is to quantify over X *internally* to the first part of the binary quantificational structure. That is, we might try something along the lines of

Re-Modified Schema
$[\![\Gamma x\ (A; B)]\!]^{i\ o}$ iff $\underline{\Gamma}\alpha\ (\exists g\ [\![A]\!]^{i^{\alpha}_{x}\ g}\ ;\ [\![B]\!]^{Y\ Z})$

In this case, our meta-language quantifier $\underline{\Gamma}$ quantifies only over one object (in our example, this will be men that are donkey-owners) but the output context for A can still vary as the input context varies.

My objection to this manoeuvre is that it destroys the prospect of genuinely *threading* information between the two arguments to the quantifier. The Output context g for the first argument cannot be threaded into the second argument. It is limited by the scope of the existential quantifier over g and that scope does not reach the second argument. If we want to secure anaphoric relations by the use of threading between the two arguments of a binary quantificational structure, then any solution using this procedure will be no solution at all.

Are there any other possibilities? Given the fact that we want to employ binary quantifiers in the meta-language, there are not, in fact, too many options. We have seen that quantifying over X with α by $\underline{\Gamma}$ is not good enough, and we have seen that quantifying over X with a quantifier whose scope only covers the first argument A is not good enough. Furthermore, there is no possibility of a quantifier whose scope extends over A and B but whose scope is narrower than $\underline{\Gamma}$. This is because $\underline{\Gamma}$ is, by hypothesis, a *binary* quantifier. We also cannot quantify over X by a quantifier with wider scope than $\underline{\Gamma}$ because X could not then vary with W within the scope of $\underline{\Gamma}$. This is not, of course, a conclusive argument that some other way to thread single assignments and quantify correctly will not be devised, but it is highly suggestive.

In fact, it is variations on the second solution above (the 'non-solution') that have been proposed by most recent contributions to the literature on discourse anaphora and the proportion problem ([Chierchia 88],[Kamp & Reyle *forth.*],[Gawron & Peters 90]).

I shall demonstrate how the first two of these accounts make use of an extra operation to account for relations between variables and quantified antecedents in binary quantificational structures, and then proceed to give a new dynamic logic with binary quantificational structures, which does genuinely thread the output from the first argument A into the second B and also does not fall foul of the proportion problem.

6.3 Recent Accounts of Dynamic Binary Quantifiers

In this section, we first review some salient points of the original presentation of DRT [Kamp 81] which did not contain binary quantifiers. Then, we examine the treatment of binary quantifiers in [Kamp & Reyle *forth.*] and a treatment of binary quantifiers in dynamic logic proposed by [Chierchia 88].

It will be found that these accounts do not maintain the uniform account of anaphora which formed one of the central motivations for DRT.

6.3.1 DRT without binary quantifiers

Three of the central motivations for the particular version of DRT as presented in [Kamp 81] are

i) that indefinites function uniformly by introducing discourse referents into a DRS
ii) that anaphoric pronouns function uniformly by looking to an *accessible* discourse referent in the DRS
iii) that the quantificational force associated with an indefinite depends on the structure of the DRS in which it is processed

These three points are illustrated in the following discourses. In each case, the truth condition of the discourse as given by the theory is shown below the sentence.

(67) a Pedro owns a donkey. He feeds it.
There is a donkey which Pedro both owns and feeds.

b If Pedro owns a donkey, he feeds it.
Every donkey which Pedro owns, he feeds.

c Every farmer courts a widow who admires him
For every farmer, there is a widow who admires him and that he courts

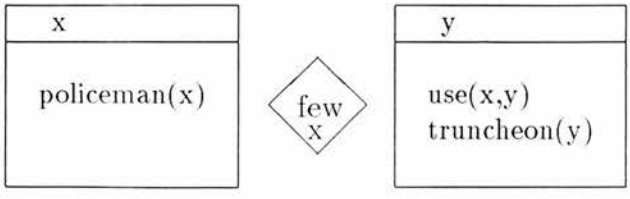
In the processing of these three discourses, the indefinite noun phrases uniformly introduce discourse referents and the pronouns are replaced uniformly by accessible discourse referents. Owing to the structure of the DRSs in which the indefinites are processed, the indefinite in 67a ends up taking existential force whilst that in 67b takes universal force. Also, the male pronoun in 67c is a classic 'bound' anaphoric pronoun and that in 67a is a classic 'discourse' pronoun - but both cases are treated identically, in DRT.

6.3.2 DRT with binary quantifiers

[Kamp & Reyle *forth.*] add binary quantifiers to DRT. These structures are called there *duplex* conditions and are intended primarily to handle such quantifiers as 'most' and 'few' - that is, those quantifiers which demand a binary quantificational analysis. Duplex conditions can be added to our definition of DRT (2.1.2) by the following clause:

Duplex Conditions
if K_1 and K_2 are DRSs confined to V and R,
then Q x $(K_1; K_2)$ is a DRS condition (Q $\in$ {most, few})

The box-notation Kamp and Reyle define for these structures is illustrated in the following DRS for 'few policeman use a truncheon'



The important point to note about these structures is that, by associating a discourse referent directly with the quantifier in the diamond, one is signalling quantification over

single objects. By contrast, the verification condition for $K_1 \Rightarrow K_2$ (the original DRS condition for 'every') induces quantification over assignments g that are extensions (of f) covering the domain U_{K_1}. In this way, one effectively quantifies over n-tuples of objects. The notation for duplex conditions indicates a return to a simpler form of quantification. The difference between the two corresponds to the difference between the Modified Schema and the Re-Modified Schema of 6.2.1 above. Unsurprisingly, the Proportion Problem besets the n-tuple treatment of DRT just as it beset our Modified Schema in dynamic logic. Indeed, the problem arose originally from the observation that the DRT treatment of 'every' via $\Rightarrow$ would not extend naturally to a quantifier such as 'most' (*cf.* [Partee 83]).

If the new duplex conditions correspond to our Re-Modified Schema then we should expect the same problems here that we faced there. The problem will be to ensure that a discourse referent y occurring in K_2 is semantically linked to another occurrence of y in K_1, even when y is not mentioned in the diamond of a DRS duplex condition.

[Kamp & Reyle *forth.*] consider two candidate verification conditions for duplex conditions: that is, two definitions of $[\![\ Qx\ (K_1; K_2)\ ]\!]^f$. First we define two sets S_1 and S_2 (with respect to f) to be associated with K_1 and K_2.

$$\begin{array}{rcl} S_1 & = & \{\alpha \mid \exists g\ (g \supseteq_{U_{K_1}} (f \cup \{< x, \alpha >\})\ \&\ [\![\ K_1\ ]\!]^g)\} \\ S_2 & = & \{\alpha \mid \exists g\ (g \supseteq_{U_{K_1}} (f \cup \{< x, \alpha >\})\ \&\ [\![\ K_1\ ]\!]^g)\ \& \\ & & \exists h\ (h \supseteq_{U_{K_2}} g\ \&\ [\![\ K_2\ ]\!]^h)\} \end{array}$$

Secondly, these two sets are compared using methods familiar from Generalized Quantifiers. That is, if 'Q' is 'most', then we check to see if $\mid S_1 \cap S_2 \mid > \frac{1}{2} \mid S_1 \mid$, if 'Q' is 'every' we check whether $\mid S_1 \cap S_2 \mid = \mid S_1 \mid$ and so on. So, if 'Q' is a natural language quantifier and Q' is its Generalized Quantifier counterpart, then

Verification of Duplex Conditions
$[\![\ Qx\ (K_1; K_2)\ ]\!]^f$ iff $Q'(S_1, S_2)$

For a sentence such as 'Every man who owns a donkey feeds it' this definition yields the truth condition that every man who owns a donkey is a man who owns and feeds a donkey. However, [Kamp & Reyle *forth.*] disapprove of this truth condition and

therefore propose the following alternative definition for the set S_2

$$\begin{aligned} S_2' = \{\alpha \mid\ & \exists g\ (g \supseteq_{U_{K_1}} (f \cup \{<x, \alpha>\}) \ \&\ [\![K_1]\!]^g)\ \& \\ & \forall g\ ((g \supseteq_{U_{K_1}} (f \cup \{<x, \alpha>\}) \ \&\ [\![K_1]\!]^g = 1) \\ & \quad \rightarrow \exists h\ (h \supseteq_{U_{K_2}} g\ \&\ [\![K_2]\!]^h = 1)\} \end{aligned}$$

The result of this definition is that 'Every man who owns a donkey feeds it' is true just in case every man who owns a donkey feeds every donkey he owns. The new definition therefore gives the same results as the earlier version of DRT without binary quantifiers, only the new version now uses duplex conditions.

Both definitions however make essential use of a *double evaluation* of the first part of the duplex condition K_1. K_1 is evaluated both in calculating the set S_1 and in calculating the set S_2. In our example, to calculate the set S_1 one first evaluates 'man that owns a donkey' in order to count the number of men that own donkeys. Then, to calculate S_2 one re-evaluates 'man that owns a donkey' in order to provide possible antecedents for anaphoric pronouns in 'feeds it'. The indefinite 'a donkey' is therefore evaluated twice in order to perform its two functions : contributing to the meaning of 'man that owns a donkey', and contributing as an antecedent to the pronoun in 'feeds it'.

The important point about the double evaluation is that it makes the treatment of anaphora across the two parts of a binary quantifier rather different from sentence-external anaphora. The interpretation of 'A man came in. He sat down' does not have to proceed by first evaluating 'A man came in' and then evaluating both 'A man came in' and 'He sat down' together in order to secure the anaphoric dependency. The fact that 'He sat down' is to be evaluated in the context created by 'A man came in' is written into the DRS construction algorithm. The generation of such dynamic effects is, in fact, the *point* of that algorithm. It would therefore appear to be a signal that something is wrong that one has to start writing additional contextual influences via re-evaluation of DRSs into the model-theoretic interpretation of the DRSs.

6.3.3 Chierchia's Dynamic Binary Quantifiers

Chierchia raises the question whether quantifiers in dynamic logic are *conservative* or not ([Chierchia 88]). In Generalized Quantifier Theory, it is a plausible semantic universal that all natural language quantifiers Q are conservative; that is, they possess the property that $Q(A;B) \leftrightarrow Q(A;A \wedge B)$. Is this true of dynamic quantifiers?

Prima facie, the question does not even make sense until one has first added binary quantifiers to dynamic logic. The question simply cannot be asked of the *unary* quantifiers of DPL, or even those of Dynamic Intensional Logic (DIL) used in Dynamic Montague Grammar (see [Groenendijk & Stokhof 91a]). Similarly, one cannot ask whether the quantifiers in Montague's IL are conservative or not. However, one can use λ-abstraction to construct an IL translation of the English quantifier 'every' which takes two arguments, such as $\lambda P \lambda Q\ \forall x(P(x) \rightarrow Q(x))$ (though this device cannot be used for quantifiers which require an *essentially* binary analysis). Call this translation **every**. One may then ask whether the following holds true: $\mathbf{every}(A)(B) \leftrightarrow \mathbf{every}(A)(A \wedge B)$. The answer is that it does. It is in a similar sense that Chierchia asks whether conservativity holds true in dynamic logic. Suppose one has added λ-abstraction to DPL and constructed a translation for 'every' : $\mathbf{every}_d$. Suppose too we let $\wedge_d$ mean dynamic conjunction (that is, DPL conjunction) then one can ask: is it the case that $\mathbf{every}_d(A)(B) \leftrightarrow \mathbf{every}_d(A)(A \wedge_d B)$? In dynamic logic however, this statement of conservativity does not hold true. The reason is not hard to find. Consider the following two statements

(68) a Every man that owns a donkey feeds it
b Every man that owns a donkey is a man that owns a donkey and feeds it

DPL has been designed to make similar predictions to DRT and therefore it predicts that 68a is true just in case every donkey-owning man feeds *every* donkey he owns. The addition of λ-abstraction to DPL does not alter this prediction. DPL also predicts that 68b is true just in case every donkey-owning man feeds at least one of the donkeys

he owns. Therefore 68a and 68b are not equivalent and neither are their DPL+λ-abstraction translations. The difference between the two readings again corresponds to the different definitions S_2 and S_2' in 6.3.2 above and between the Modified and Re-Modified Schemata of 6.2.1.

Rather than conclude that 'every' and **every**$_d$ are not conservative (in the sense just stated) in a dynamic environment, Chierchia suggests that the *proper* translations of English quantifiers are not actually given by DPL in the first place. He therefore defines some new quantifier symbols by the following schema

Q^+ Schema
$Q^+(A)(B) =_{\text{def}} \mathrm{Q}(A)(A \wedge_d B)$

and claims that instances of Q^+ are good translations for natural language quantifiers. That is, in English, 68b actually is equivalent to 68a, and therefore quantifiers are (dynamically) conservative, after all. The empirical issue about English on which this matter depends is discussed further below (6.5).

It is rather pleasing to be able to rewrite Chierchia's claim without using the detour through λ-abstractions and the Q^+ **Schema**. The correspondences we have set up between our schemata (Modified and Re-Modified), the definitions S_2 and S_2' and the readings of 68a and 68b suggest that the Q^+ **Schema** should be closely related to our Re-Modified Schema from 6.3.2. It is. It is an instance of it.

First Instance of Re-Modified Schema
$[\![\, \Gamma x\ (A; B) \,]\!]^{i\ o}$ iff $i = o \wedge \underline{\Gamma}\alpha\ (\exists g\ [\![\, A \,]\!]^{i^{\alpha}_{x}\ g}\ ;\ \exists h\ ([\![\, A \,]\!]^{i^{\alpha}_{x}\ h} \wedge \exists j\ ([\![\, B \,]\!]^{h\ j})))$

This instance corresponds to Chierchia's 'closed' quantifiers - closed in the sense that dynamic effects are not passed on to subsequent sentences. A similar version can be given for the 'open' quantifiers.

There is an interesting further twist to this tale. If we look back at our definitions of conjunction and truth in 2.2, we find that the First Instance can be re-written more simply as

$[\![\, \Gamma x\ (A; B) \,]\!]^{i\ o}$ iff $i = o \wedge \underline{\Gamma}\alpha\ (A$ is true w.r.t. $\mathrm{i}^{\alpha}_{\mathrm{x}}; A \wedge B$ is true w.r.t. $\mathrm{i}^{\alpha}_{\mathrm{x}})$

What more could one ask for from a conservative dynamic generalized quantifier!

(In a recently published article, [Chierchia 92], Chierchia claims the above result provides one reason why his account is not just a technical variant on a DRT based theory which copies material from the left-hand DRS to the right-hand one within a duplex condition. The reason is that his theory embodies an empirical claim - since 'weak' readings are linked to the property of conservativity, which is a semantic universal, so his theory claims that no language can disallow the 'weak' readings for relative clause donkey anaphora. However, since the 'linking' relation is not entailment, the empirical claim for donkey anaphora is just an additional claim over and above that of conservativity for natural language determiners. In this sense, DRT could presumably add a similar extra empirical claim - duplex conditions will always be evaluated using a double evaluation strategy (in the syntax or the semantics). Nevertheless, the similarity to conservativity remains intriguing).

Nevertheless, there are two reasons for thinking that the approach embodied by the Q^+ **Schema** and the **First Instance** is unsatisfactory. The first is just the not unreasonable worry that the procedure looks a little too close to an analysis which simply rewrites $Q(A; B)$ to $Q(A; A \wedge B)$ in the syntax before beginning any evaluation at all.

The second reason, and the more important one for our purposes, is that Chierchia's dynamic binary quantifiers, like our **First Instance**, make use of double evaluation. The information derived from the first argument to the binary quantifier is not *threaded* into the second argument but simply copied there by another evaluation. (It is threaded within the second argument by dynamic conjunction but our current concern is relating the arguments, not what goes on inside them). The second evaluation is present precisely so that possible anaphors in B can be related to their antecedents in A - but this very possibility was the reason why we were interested in the concept of threading in the first place. In the next section I will develop a logic for binary quantificational

structures which does thread the output context derived from processing A directly into the input for B. There will be no evaluating of A twice.

6.4 The System $\mathrm{DPL}_{\mathrm{bq}}$

I shall now present a dynamic semantics for an extension of DPL that contains binary quantifiers and genuinely threads information between the two arguments to a quantifier. The language is named $\mathrm{DPL}_{\mathrm{bq}}$.

In DPL, the evaluation of a formula such as $man(x) \wedge \exists y\ (donkey(y) \wedge owns(x,y))$ relates an input context to a *number* of output contexts each of which determines in the yth position some particular donkey that is owned by the value of the variable x. As we noted above (6.2.1), if one man named 'j' owns two donkeys ('p' and 'q') then, for an input assignment $\emptyset^{j}_{x}$, there are two different output assignments $\emptyset^{jp}_{xy}$ and $\emptyset^{jq}_{xy}$ which together with $\emptyset^{j}_{x}$ will satisfy our formula. However, we found that threading such assignments almost inevitably led to the Proportion Problem and our mis-counting. For the purposes of quantification we wanted to count donkey-owning men, but, for the purposes of threading, we wanted to quantify over the differing output assignments.

The proposal in $\mathrm{DPL}_{\mathrm{bq}}$ is to make our input and output contexts not single assignments but *sets* of assignments. The fundamental idea is that, whereas our earlier output assignments recorded per man, some particular donkey that he owned, the new output contexts will determine the set of all the donkeys that he owns. Given an input set of assignments $\{\emptyset^{j}_{x}\}$, for example, the output set will be $\{\emptyset^{jp}_{xy}, \emptyset^{jq}_{xy}\}$. Since there is a one-to-one correspondence between men that own donkeys and the sets of donkeys that they own, we will not be led to miscount. However, we can still thread information concerning which donkeys each man owns into evaluations of subsequent formulae. In this way, we will avoid the Proportion Problem while still threading anaphoric information.

There are two further important new aspects of the system of $\mathrm{DPL}_{\mathrm{bq}}$.

Selecting from a Context

The first aspect is that, since we are going to be threading sets of assignments between formulae, we will need to have some method of *selecting* one of them from a set to give the value of a subsequent variable. Suppose we have the discourses 69a,b with their $\mathrm{DPL_{bq}}$ formalizations as indicated.

(69) a John owns *a donkey. It* is well-fed.
$\exists y\ (d(y); o(j,y)) \wedge w(y)$
b John owns *a donkey.* Every sheep distrusts *it.*
$\exists y\ (d(y); o(j,y)) \wedge \forall x\ (s(x); d(x,y))$

The general idea is that, in each case, the first sentence of the discourse will generate an output set of assignments determining, via the yth value of each assignment in the set, all the donkeys that John owns. Also, in each case, the value of 'it' in the second sentence should be *one* of the donkeys that John owns. Therefore, at some point we will need to select one of the members of the input set of assignments to provide the value for 'it'. However, we must ensure that in 69b this selection occurs outside the scope of 'every sheep' - otherwise we will end up paraphrasing 69b by 'John owns a donkey and every sheep distrusts a donkey that John owns', and that is not what the sentence means.

The method we adopt for securing this result is a mutual recursion on two semantic notions. The definition of DPL proceeded by a direct recursion on 'true with respect to a pair of assignments'. That notion occurred on both the left and right hand sides of the clauses of our semantic rules. The definition of $\mathrm{DPL_{bq}}$ will recurse through two similar relative notions: $\mathrm{true_1}$ and $\mathrm{true_2}$ (abbreviated to T_1 and T_2). In particular, the semantic rule associated with each syntactic construction will be a T_2-rule. That is, each will define when a construction is $\mathrm{true_2}$ and it will do so in terms of when the parts of the construction are $\mathrm{true_1}$. There will only be one T_1-rule and this will define $\mathrm{true_1}$ in terms of $\mathrm{true_2}$. The mutual recursion will terminate with a base case T_2-rule.

The general procedure is as follows. Given a formula and an input context, we first use the T_1-rule in order to select a suitable member of the input context. This will have

the effect of 'binding' any free variables in the formula. Free variables such as 'it' in 69b will thereby be assigned a value outside the scope of any other quantifiers in the formula containing them. Having done this, we then recurse on a T_2-rule which will break-down the formula into its constituent parts according to normal practice. We will then recurse again on the T_1 rule in order to bind any *newly freed* variables. This is necessary to deal with cases such as 70a, formalized as shown.

(70) a John owns *a donkey*. Every sheep that ate a turnip hid it from it.
$\exists y\ (d(y); o(j, y)) \wedge \forall x\ (s(x) \wedge \exists z\ (t(z); a(x, z)); h(x, z, y))$

In this example, there may be different turnips per sheep (each sheep hides a different turnip) but there are not different donkeys per sheep (all the sheep are being secretive towards the same donkey(s)).

This is reflected in the formal syntax by the fact that, whilst there are no free variables in the discourse as a whole, y is free in the formalization of the second sentence. y must take its value from the input context for the second sentence *before* we start evaluating the universal quantifier. At the level of the second sentence, however, z is not free and cannot be assigned a value. Only after we have decomposed the universally quantified formula into its two parts, thereby 'freeing' z in the second part, can we recursively call the T_1-rule so that z is assigned a value.

Naming Contexts

Now that contexts are sets of assignments it will be the case that for any given formula ϕ and input context i there is only one output context o such that i and o satisfy ϕ. Given a discourse consisting of a sequence of sentences $S_0, S_1, \ldots, S_n$ and a particular initial context c_0, it follows that the sequence of contexts generated by the discourse will be uniquely determined. That is, if we evaluate S_0 in c_0, there is a unique output context c_1, and if evaluate S_1 in c_1 a further uniquely determined context is generated. In this situation, we can simplify our meta-language greatly by *identifying* input and output contexts by the particular occurrences of formulae whose contexts

they are. To achieve this, I add unique indices to each formula and sub-formula in a discourse. Then, to refer to the input context of the particular formula indexed by i, use $I(i)$ and to refer to i's output context use $O(i)$.

6.4.1 $\mathrm{DPL_{bq}}$ Formal Details

$\mathrm{DPL_{bq}}$ Syntax

$\mathrm{DPL_{bq}}$ consists of a set of non-logical constants, variables and n-ary predicates, together with the logical constants: $\neg, \wedge, \vee, \rightarrow$ plus quantifiers of existence ($\exists$) , universality ($\forall$) and preponderance (Most). Also, we include the natural numbers $\mathcal{N}$ which will serve to label individual occurrences of formulae: ϕ_i will be the (pre-)formula ϕ labelled with the number i.

The set of pre-formulae of $\mathrm{DPL_{bq}}$ is given by the following rules, where i, j, k are variables over $\mathcal{N}$: if $t_1, \ldots, t_n$ are variables or non-logical constants and P is an n-ary predicate, then $P(t_1, \ldots, t_n)_i$ is a pre-formula; if ϕ is a pre-formula, then so is $\neg(\phi_i)_j$; if ϕ and ψ are pre-formulae then so are $(\phi_i \wedge \psi_j)_k$, $(\phi_i \vee \psi_j)_k$,$(\phi_i \rightarrow \psi_j)_k$; if ϕ and ψ are pre-formulae and x is a variable, then $\exists x\ (\phi_i; \psi_j)_k$, $\forall x\ (\phi_i; \psi_j)_k$, and Most $x(\phi_i; \psi_j)_k$ are pre-formulae too.

A $\mathrm{DPL_{bq}}$ formula is a pre-formula where no label occurs more than once.

Free variables

In the following definition, μ and ν range over object language variables and, for convenience, I omit labels.

1. If ϕ is of form $P(t_1, \ldots, t_n)$,
   $FV(\phi)$ is the set of all variables occurring in ϕ
   $AQ(\phi) = \emptyset$.
2. If ϕ is of form $\neg(\psi)$,
   $FV(\phi) = FV(\psi)$
   $AQ(\phi) = \emptyset$
3. If ϕ is of form $(\psi \vee \pi)$,
   $FV(\phi) = FV(\psi) \cup FV(\pi)$.
   $AQ(\phi) = \emptyset$
4. If ϕ is of form $(\psi \wedge \pi)$,
   $FV(\phi) = FV(\psi) \cup \{\mu \mid \mu \in FV(\pi), \mu \notin AQ(\psi)\}$
   $AQ(\phi) = AQ(\psi) \cup AQ(\pi)$
5. If ϕ is of form $(\psi \rightarrow \pi)$,
   $FV(\phi) = FV(\psi) \cup \{\mu \mid \mu \in FV(\pi), \mu \notin AQ(\psi)\}$
   $AQ(\phi) = \emptyset$
6. If ϕ is of form $\exists\mu(\psi; \pi)$,
   $FV(\phi) = (FV(\psi) \cup \{\nu \mid \nu \in FV(\pi), \nu \notin AQ(\psi)\}) - \{\mu\}$
   $AQ(\phi) = AQ(\psi) \cup AQ(\pi) \cup \{\mu\}$
7. If ϕ is of form $\forall\mu(\psi; \pi)$ or $\text{Most}\mu(\psi; \pi)$,
   $FV(\phi) = (FV(\psi) \cup \{\nu \mid \nu \in FV(\pi), \nu \notin AQ(\psi)\}) - \{\mu\}$
   $AQ(\phi) = \emptyset$

A DPL_{bq} discourse is a formula with no free variables.

DPL_{bq} Semantics

Models for DPL_{bq} resemble those for DPL. Each is a pair $< D, F >$ where D is a non-empty set of individuals and F is a function from non-logical constants to members of D, and from n-ary predicates to n-ary relations on D. Assignments are partial functions from variables to members of D. A *context* is a set of assignments. Let G be the set of all contexts and L_σ be the set of labels for a DPL_{bq} discourse σ. I_σ and O_σ are functions from L_σ to G, for σ. For any label j, $I_\sigma(j)$ is the *input context* for the formula labelled with j and $O_\sigma(j)$ is that formula's *output context.* The rules for determining an output context on the basis of the input context and a formula are given below as the Context Change Function. When the discourse σ is obvious, I omit the subscript σ in I_σ and O_σ.

As usual, g_x^α is the assignment g extended or modified by assigning α to x. If k and g are assignments such that $\exists\alpha(k = g_x^\alpha)$ then we write $k \overset{x}{\sim} g$. Furthermore, if j and k

are partial assignments then we write $j \supset k$ if j and k agree on all variables on which k is defined. Trivially, $k \supset k$.

If i is a set of assignments then $i_x^+ = \{f \mid \exists j(j \in i \wedge f \overset{x}{\sim} j\}$. That is, i_x^+ is the set of assignments each of whose members is just like a member of i except that it extends or modifies that member by the value assigned to x.

For individual constants and variables, we define $[\![x]\!]^g$ (the value of x relative to g) to be $g(x)$ for all variables x ; $[\![c]\!]^g$ is $F(c)$ for all constants c.

Context Change Function (CCF)

1. If $\sigma = P(t_1, \ldots, t_n)_i$, then
   $O(i) = \{f \in I(i) \mid < [\![t_1]\!]^f, \ldots, [\![t_n]\!]^f > \in F(P) \}$
2. If $\sigma = (\phi_i \wedge \psi_j)_k$ then,
   $I(i) = I(k)$
   $O(i) = I(j)$
   $O(j) = O(k)$
3. If $\sigma = (\phi_i \vee \psi_j)_k$ then,
   $I(i) = I(k)$
   $I(j) = I(k)$
   $O(k) = \{f \in I(k) \mid f \in O(i) \vee f \in O(j)\}$
4. If $\sigma = (\phi_i \rightarrow \psi_j)_k$ then,
   $I(i) = I(k)$
   $O(i) = I(j)$
   $O(j) = O(k)$
5. If $\sigma = \neg(\phi_i)_j$ then,
   $I(j) = I(i)$
   $O(j) = \{f \in I(j) \mid f \notin O(i)\}$
6. If $\sigma = \exists x(\phi_i; \psi_j)_k$, then
   $I(i) = I(k)_x^+$
   $O(i) = I(j)$
   $O(j) = O(k)$
7. If $\sigma = \forall x(\phi_i; \psi_j)_k$, then
   $I(i) = I(k)_x^+$
   $O(i) = I(j)$
   $O(k) = \{f \in I(k) \mid \forall \alpha \ (f_x^\alpha \in O(i); f_x^\alpha \in O(j))\}$
8. If $\sigma = \text{Most}x(\phi_i; \psi_j)_k$, then
   $I(i) = I(k)_x^+$
   $O(i) = I(j)$
   $O(k) = \{f \in I(k) \mid \text{Most } \alpha(f_x^\alpha \in O(i); f_x^\alpha \in O(j))\}$

We shall make use of two subsidiary notions: truth_1 of a formula ϕ with respect to an assignment g, written $[\![\phi]\!]_{T_1}^g$ and truth_2 of a formula ϕ with respect to an assignment g, written $[\![\phi]\!]_{T_2}^g$.

A DPL_{bq} discourse is *true* just in case it is true_1 with respect to the empty assignment where the input context is $\{\emptyset\}$. That is, if i is the label of the whole discourse ϕ then 'ϕ is *true* iff $[\![\phi]\!]^{\emptyset}_{T_1}$', where $I_\phi(i) = \{\emptyset\}$.

T_1-rule

T_1 $[\![\phi_i]\!]^g_{T_1}$ iff $\exists f(f \supset g \wedge f \in I(i) ; [\![\phi_i]\!]^f_{T_2})$

T_2-rules

1. $[\![P(t_1, \ldots, t_n)_i]\!]^g_{T_2}$ iff $g \in O(i)$
2. $[\![(\phi_i \wedge \psi_j)_k]\!]^g_{T_2}$ iff $[\![\phi_i]\!]^g_{T_1} \wedge [\![\psi_j]\!]^g_{T_1}$
3. $[\![(\phi_i \vee \psi_j)_k]\!]^g_{T_2}$ iff $[\![\phi_i]\!]^g_{T_1} \vee [\![\psi_j]\!]^g_{T_1}$
4. $[\![(\phi_i \rightarrow \psi_j)_k]\!]^g_{T_2}$ iff $[\![\phi_i]\!]^g_{T_1} \rightarrow [\![\psi_j]\!]^g_{T_1}$
5. $[\![\neg(\phi_i)_j]\!]^g_{T_2}$ iff $\neg [\![\phi_i]\!]^g_{T_1}$
6. $[\![\exists x \ (\phi_i; \psi_j)_k]\!]^g_{T_2}$ iff $\exists \alpha \ ([\![\phi_i]\!]^{g^\alpha_x}_{T_1}; [\![\psi_j]\!]^{g^\alpha_x}_{T_1})$
7. $[\![\forall \ x(\phi_i; \psi_j)_k]\!]^g_{T_2}$ iff $\forall \ \alpha([\![\phi_i]\!]^{g^\alpha_x}_{T_1}; [\![\psi_j]\!]^{g^\alpha_x}_{T_1})$
8. $[\![\text{Most } x(\phi_i; \psi_j)]\!]^g_{T_2}$ iff $\text{Most } \alpha([\![\phi_i]\!]^{g^\alpha_x}_{T_1}; [\![\psi_j]\!]^{g^\alpha_x}_{T_1})$

Discussion

Having made contexts into sets of assignments, DPL_{bq} has separated out the rules for context change from those giving truth-conditional content in an elegant way (*cf.* the complexity invoked by [Schubert & Pelletier 89] who also separate the two notions). Furthermore, the **T_2-rules** have reverted to a system where each object-language operation is analyzed via use of the equivalent meta-language operation (where the meta-language is still standardly interpreted). For example, whereas DPL defines dynamic conjunction via 71a, DPL_{bq} uses 71b.

(71) a $[\![(\phi \wedge \psi)]\!]^{i \ o}$ iff $\exists k \ ([\![\phi]\!]^{i \ k}$ *and* $[\![\psi]\!]^{k \ o})$

b $[\![(\phi_i \wedge \psi_j)_k]\!]^g_{T_2}$ iff $[\![\phi_i]\!]^g_{T_1} \wedge [\![\psi_j]\!]^g_{T_1}$

The extra complication of context change in DPL_{bq} has been written into the separate definitions of the Context Change Function. In the case of conjunction (clause 2) all

we need to do is let the input for the whole conjoined formula be the input to the first conjunct, let the output for the first conjunct become the input to the second conjunct and let the output for the second conjunct be the output for the conjunction.

As another example, the DPL_{bq} rule for universal quantification is quite standard : the analysis of an object language binary structured quantifier is given by the *use* of a (standardly interpreted) corresponding meta-language binary structured quantifier.

(72) a. $[\![\forall x(\phi_i; \psi_j)_k]\!]^g_{T_2}$ iff $\forall \alpha([\![\phi_i]\!]^{g^\alpha_x}_{T_1}; [\![\psi_j]\!]^{g^\alpha_x}_{T_1})$

In investigating the dynamic effects of DPL_{bq} it will be the rules for context change that will be of primary interest. The rule for conjunction we have explained. The rule for existential quantification is similar except that the input context to the first argument of the quantifier is an expansion of the input context to the whole formula. Just as one ordinarily extends (or modifies) a partial *single* assignment to deal with the newly quantified variable, so the input *set* of assignments is expanded in a similar fashion in DPL_{bq} . The rules for $\forall x$ and Most x also expand the input context to cover the new variable x and they also thread the output context for the first argument to the quantifier into the input for the second. However, the output contexts for these quantifiers are a *subset* of the input contexts so the possible values of x derived from processing within the (ordinary) scope of these two quantifiers is not passed on to subsequent formulae. This feature is designed to reflect the following judgements on possible anaphora

(73) a. A man came in. He sat down,
b. * Every man came in. He sat down.
c. * Most men came in. He sat down.

The fact that for all three quantifiers the output context for the first argument is threaded into the second is designed to reflect the following anaphoric possibilities

(74) a. A man that owned a donkey fed it.
b. Every man that owned a donkey fed it.
c. Most men that owned a donkey fed it.

6.4.2 Examples

In deriving a truth condition for a formula of $\mathrm{DPL}_{\mathrm{bq}}$, one must keep track of two things. First, we will be alternately applying a **T_1-rule** and a **T_2-rule**, because of the mutual recursion in the definitions. Secondly, whenever we apply a **T_2-rule** we will also have to apply the appropriate rule for Context Change. We will present a derivation in a table containing four columns, where the first column contains a name for the current row, the second contains the name of the semantic rule to be applied, the third contains the truth conditional content derived by applying the named rule to the content of the previous line and the fourth contains the context conditions derived from applying the corresponding context change rule.

Although *every* application of a **T_2-rule** causes a recursion on the **T_1-rule** rather than directly recursing on another **T_2-rule**, it is often the case that the detour via the **T_1-rule** has no significant effect. The reason, as explained above, is that it is the job of the **T_1-rule** to 'mop-up' newly freed variables created by the deconstruction of formulae. If there are no such variables then the **T_1-rule** has no work to do. As an example, consider any discourse ϕ_i to be processed in the initial context $\{\emptyset\}$. The rule for truth tells us to evaluate $[\![\ \phi_i\ ]\!]^{\emptyset}_{T_1}$ where $I(i) = \{\emptyset\}$. The **T_1-rule** tells us to find some assignment f which extends $\emptyset$, is a member of $\{\emptyset\}$ and which has the property that $[\![\ \phi_i\ ]\!]^{f}_{T_2}$. Since there is only *one* assignment which extends $\emptyset$ and is a member of $\{\emptyset\}$, namely $\emptyset$ itself, we can deduce that $[\![\ \phi_i\ ]\!]^{\emptyset}_{T_1}$ holds just in case $[\![\ \phi_i\ ]\!]^{\emptyset}_{T_2}$. In the example derivations that follow, whenever an application of the **T_1-rule** has no significant effect on free variables, I shall simply rewrite $[\![\ \phi_i\ ]\!]^{g}_{T_1}$ to $[\![\ \phi_i\ ]\!]^{g}_{T_2}$ without comment.

First Example - $\exists x\ (m(x); c(x)) \wedge \neg(s(x)))$

The first example is intended to represent the discourse 75

(75) A man came in. He did not sit down.

First, we will uniquely index all the subformulae. The result is 76.

(76) $(\exists x\ (m_3(x); c_4(x))_2 \wedge \neg(s_6(x))_5)_1$

	Truth Conditional Content	Context Conditions
	$[\![\exists x\ (m_3(x); c_4(x))_2 \wedge \neg(s_6(x))_5)_1]\!]^{\emptyset}_{T_1}$ iff ...	$I(1) = \{\emptyset\}$
a) [T_1]	$[\![(\exists x\ (m_3(x); c_4(x))_2 \wedge \neg(s_6(x))_5]\!]^{\emptyset}_{T_2}$	
b) [2]	$[\![\exists x\ (m_3(x); c_4(x))_2]\!]^{\emptyset}_{T_1} \wedge [\![\neg(s_6(x))_5]\!]^{\emptyset}_{T_1}$	$I(2) = \{\emptyset\}$, $O(2) = I(5)$, $O(5) = O(1)$
c) [T_1]	$[\![\exists x\ (m_3(x); c_4(x))_2]\!]^{\emptyset}_{T_2} \wedge [\![\neg(s_6(x))_5]\!]^{\emptyset}_{T_1}$	
d) [6]	$\exists\alpha\ ([\![m_3(x)]\!]^{\emptyset^{\alpha}_{x}}_{T_1}; [\![c_4(x)]\!]^{\emptyset^{\alpha}_{x}}_{T_1}) \wedge [\![\neg(s_6(x))_5]\!]^{\emptyset}_{T_1}$	$I(3) = \{g \mid g \overset{x}{\sim} \emptyset\}$, $O(3) = I(4)$, $O(4) = O(2)$
e) [T_1]	$\exists\alpha\ ([\![m_3(x)]\!]^{\emptyset^{\alpha}_{x}}_{T_2}; [\![c_4(x)]\!]^{\emptyset^{\alpha}_{x}}_{T_1}) \wedge [\![\neg(s_6(x))_5]\!]^{\emptyset}_{T_1}$	
f) [1]	$\exists\alpha\ (\alpha \in F(m); [\![c_4(x)]\!]^{\emptyset^{\alpha}_{x}}_{T_1}) \wedge [\![\neg(s_6(x))_5]\!]^{\emptyset}_{T_1}$	$O(3) = \{g \mid g \overset{x}{\sim} \emptyset, g(x) \in F(m)\}$
g) [T_1]	$\exists\alpha\ (\alpha \in F(m); [\![c_4(x)]\!]^{\emptyset^{\alpha}_{x}}_{T_1}) \wedge [\![\neg(s_6(x))_5]\!]^{\emptyset}_{T_2}$	
h) [1]	$\exists\alpha\ (\alpha \in F(m); \alpha \in F(c)) \wedge [\![\neg(s_6(x))_5]\!]^{\emptyset}_{T_1}$	$O(4) = \{g \mid g \overset{x}{\sim} \emptyset, g(x) \in F(m), g(x) \in F(c)\}$

At this point, we have finished processing the first sentence so let us take stock. The truth conditional content for $\exists x\ (m(x); c(x))$ has resulted, unsurprisingly, in $\exists\alpha\ (\alpha \in F(m); \alpha \in F(c))$. We have also generated an output context for our first sentence, *i.e.* $O(2)$. By step d) we know that $O(2) = O(4)$ and by step h) we know that $O(4) = \{g \mid g \overset{x}{\sim} \emptyset, g(x) \in F(m), g(x) \in F(c)\}$: that is, all assignments defined only on x whose xth values are men that came in. This is the context in which we will evaluate $[\![\neg(s_6(x))_5]\!]^{\emptyset}_{T_1}$.

The next rule to apply will be the **T_1-rule**, and this time there is an appropriate free variable which should be supplied with a value.

$[\![(\exists x\ (m_3(x); c_4(x))_2 \wedge \neg(s_6(x))_5)_1]\!]^{\emptyset}_{T_1}$ iff ...

i) [T_1] $\exists\alpha\ (\alpha \in F(m); \alpha \in F(c)) \wedge \exists h\ (h \supset \emptyset \wedge h \in I(5); [\![\neg(s_6(x))_5]\!]^{h}_{T_2} \in F(s))$

The rule tells us there must be some *extension* of $\emptyset$, call it h, in $I(5)$ such that $[\![\neg(s_6(x))_5]\!]^h_{T_2}$. Whatever h we pick, we will inevitably be assigning to x *some particular* man that came in. This is because *all* the values of x in I(5) are men that came in.

Now we continue with a sub-derivation for $[\![\neg(s_6(x))_5]\!]^h_{T_1}$.

	$[\![\neg(s_6(x))_5]\!]^h_{T_2}$ iff ...	
aa) [5]	$\neg[\![s_6(x)]\!]^h_{T_1}$	$I(5) = I(6)$, $O(5) = \{f \mid f \in I(5), f \notin O(6)\}$
bb) [T_1]	$\neg\exists m(m \supset h \wedge m \in I(6); [\![s_6(x)]\!]^m_{T_2})$	
cc) [1]	$\neg\exists m(m \supset h \wedge m \in I(6); [\![x]\!]^m \in F(s))$	$O(6) = \{f \mid f \in I(6), [\![x]\!]^f \in F(s)\}$

Once again, although the demand is that no *extension* m of h should have a certain property, in fact this amounts to nothing more than that h itself should not have that property. The reason is that h is defined only on x and so is m. The condition in cc) amounts to no more than

dd) $[\![x]\!]^h \notin F(s)$

We can now substitute the result of our sub-derivation back into our truth condition i)

$$[\![\exists x\ (m_3(x); c_4(x))_2 \wedge \neg(s_6(x))_5)_1]\!]^{\emptyset}_{T_1} \text{ iff } \ldots$$

j) [subs] $\exists\alpha\ (\alpha \in F(m); \alpha \in F(c)) \wedge \exists h\ (h \supset \emptyset \wedge h \in I(5); [\![x]\!]^h \notin F(s))$

Since $I(5)$ contains all assignments defined only on x whose xth values are men that run, step j can be rewritten as

$$\exists\alpha\ (\alpha \in F(m); \alpha \in F(c)) \wedge \exists\beta\ (\beta \in F(m) \wedge \beta \in F(c); \beta \notin F(s)))$$

which is itself equivalent to

$$\exists\alpha\ (\alpha \in F(m) \wedge \alpha \in F(c) \wedge \alpha \notin F(s))$$

The truth conditions of the whole formula are therefore that some man should both have come in and not sat down, which are the correct truth conditions for the discourse.

What is the output context for the whole formula ? We can derive the output condition starting from our knowledge concerning $I(5)$.

$I(5) = \{g \mid g \overset{x}{\sim} \emptyset, g(x) \in F(m), g(x) \in F(c)\}$
$I(6) = I(5)$ (by step h)
$O(6) = \{g \mid g \overset{x}{\sim} \emptyset, g(x) \in F(m), g(x) \in F(c), g(x) \in F(s)\}$ (context change rule 1)
$O(5) = \{f \in I(5) \mid f \notin O(6)\}$ (by step h)
$O(5) = \{g \mid g \overset{x}{\sim} \emptyset, g(x) \in F(m), g(x) \in F(c), g(x) \notin F(s)\}$ (conclusion)
$O(1) = O(5)$ (by step b)

Therefore, the output context for the whole formula $O(1)$ determines all the men who came in and did not sit down. This means that a subsequent reference to 'him', will be to a man that came in and did not sit down. This is the correct prediction. In the discourse 'A man came in. He did not sit down. He did not take off his hat' the second 'he' should indeed refer to some man that came in but did not sit down.

Second Example - Most $x((m(x) \wedge \exists y\ (d(y); o(x,y))); f(x,y))$

The second example is the intended representation for 'Most men that own a donkey feed it'. First, we label all the (sub-)formulae as follows

(77) Most $x((m_3(x) \wedge \exists y\ (d_5(y); o_6(x,y))_4)_2; f_7(x,y))_1$

Rather than giving a full derivation of both truth and context conditions, I shall concentrate simply on how the contexts are threaded through the formula, beginning with the empty context $\{\emptyset\}$. The derivation proceeds as usual in top-down left-to-right fashion, therefore we apply the rule for 'Most' first, followed by that for conjunction and then the rule for 'm_3' and so on.

$I(1) = \{\emptyset\}$

$I(2) = \{g \mid g \overset{x}{\sim} \emptyset\}$ (rule 8:Most)

$I(3) = \{g \mid g \overset{x}{\sim} \emptyset\}$ (rule 2:$\wedge$)

$O(3) = \{g \mid g \overset{x}{\sim} \emptyset, g(x) \in F(m)\}$ (rule 1:m_3)

$I(4) = \{g \mid g \overset{x}{\sim} \emptyset, g(x) \in F(m)\}$ (rule 2:$\wedge$)

$I(5) = \{g \mid g \overset{xy}{\sim} \emptyset, g(x) \in F(m)\}$ (rule 6:$\exists$)

$O(5) = \{g \mid g \overset{xy}{\sim} \emptyset, g(x) \in F(m), g(y) \in F(d)\}$ (rule 1:d_5)

$I(6) = \{g \mid g \overset{xy}{\sim} \emptyset, g(x) \in F(m), g(y) \in F(d)\}$ (rule 6:$\exists$)

$O(6) = \{g \mid g \overset{xy}{\sim} \emptyset, g(x) \in F(m), g(y) \in F(d), < g(x), g(y) > \in F(o)\}$ (rule 1:o_6)

$O(4) = \{g \mid g \overset{xy}{\sim} \emptyset, g(x) \in F(m), g(y) \in F(d), < g(x), g(y) > \in F(o)\}$ (rule 6:$\exists$)

$O(2) = \{g \mid g \overset{xy}{\sim} \emptyset, g(x) \in F(m), g(y) \in F(d), < g(x), g(y) > \in F(o)\}$ (rule 2:$\wedge$)

$O(2)$ is the information that is threaded out of the first part of our binary structure and into the second part. It is a set of assignments defined only on x and y whose xth object is a man, whose yth object is a donkey and such that those two objects stand in the relation of ownership.

By rule 8, we will next apply the **T_1-rule** to the second part of the binary structure $f_7(x, y)$ with respect to the assignment $\emptyset_x^\alpha$ (where α denotes a man that owns a donkey). In this formula, both x and y are free, but only y is *newly freed.* This is reflected in the fact that our current assignment $\emptyset_x^\alpha$ is defined on x and not defined on y. So when the **T_1-rule** asks for an *extension* of $\emptyset_x^\alpha$ which is in $I(7)$, we will not be altering the current value of x but we will be selecting some particular donkey that is owned by α to be the value of y. In this way, the truth conditions for 'Most men that own a donkey feed it' are predicted to be 'most men that own a donkey feed a donkey they own.'

Third Example - $\exists\, x(m(x); \exists\, y(m(y); met(x, y))) \;\wedge\; g(x, y)$

This example is the representation for 'A man met a man. He greeted him.' Assume, as usual, that we evaluate this sentence with an input context of $\{\emptyset\}$ and with the assignment function $\emptyset$. I shall take it for granted that the output context for the first sentence is $\{g \mid g \overset{xy}{\sim} \emptyset, g(x) \in F(m), g(y) \in F(m), < g(x), g(y) > \in F(met)\}$, *i.e.* the set of all assignments defined only on x and y whose xth and yth objects are men that met each other. By the rule for conjunction, we will be evaluating the second sentence

with that same context for its input and also with the empty assignment function. Both variables are free in the second sentence and not assigned a value by the assignment function. In this instance, the **T_1-rule** will select a member of the input context and thereby assign values to both x and y. In particular, the value assigned to x will be a man that met the value of y and the value assigned to y will be a man that met the value of x. So, the whole discourse will be true just in case a pair of men that met each other also greeted each other. The discourse does *not* mean that a man that met a man greeted a man that met a man.

6.5 Evaluating DPL_{bq}

The main motivation for DPL_{bq} concerned the possibility of adding binary structured quantifiers to DPL whilst also threading information between the two arguments to the quantifier. In this way, one might hope both to gain the expressive power of generalized quantifiers in a dynamic environment and to retain the similarity of treatment of sentence-external and sentence-internal anaphora from the original presentation of DRT. The system of DPL_{bq} satisfies these two goals.

However, these two goals are somewhat theoretical ones and their achievement should not be bought at the cost of implausible empirical predictions. How does DPL_{bq} fare empirically? DPL_{bq} makes similar predictions to the system of Chierchia described earlier (6.3.3), though it makes them by a different mechanism. In particular, we can note that DPL_{bq} predicts that the following sentences' truth conditions can be paraphrased as shown

(78) a A man that owns a donkey feeds it
A man that owns a donkey feeds a donkey he owns
b Every man that owns a donkey feeds it
Every man that owns a donkey feeds a donkey he owns
c Most men that own a donkey feed it
Most men that own a donkey feed a donkey they own
d No man that owns a donkey feeds it
No man that owns a donkey feeds a donkey he owns

We have already noted that Kamp and Reyle disapprove of the 'conservative' reading for 78b. They prefer the condition that each man should feed all of the donkeys he owns. Chierchia, meanwhile, cites examples from [Schubert & Pelletier 89] such as

(79) Every man who has a dime puts it in the meter

where the idea that each man puts all his dimes in the meter seems distinctly implausible, though there are other well-known examples (*cf.* [Heim 90]) such as

(80) Every man that owned a slave owned its offspring

which seem to point to the opposite conclusion.

The distinguishing cases for the two conditions are just those where a man owns *more than one* donkey and most people's intuitions are decidedly uncertain in these cases. Similar insecurity arises with 78c. Interestingly though, intuitions appear to be rather firmer with 78a and especially with 78d. In both these cases, the predictions of $\mathrm{DPL}_{\mathrm{bq}}$ are the correct ones. In particular, the truth conditions for 78d are that no man that owns a donkey feed *any* donkey he owns. This 'strong' requirement on sentences quantified over by 'no' was stressed by [Rooth 87].

Reactions to these differences in the data differ. Rooth's reaction is that 'Since speakers have uncertain intuitions, it is not necessary or indeed appropriate to make a claim about which ...is right. Rather, a theory should provide a framework in which the range of intuitions can be modeled' ([Rooth 87]). In a similar vein, Gawron and Peters state that 'Debate has been heated ...; we simply note that whichever position turns out to be right can be captured within the framework of our current analysis' [Gawron & Peters 90]. There is something to be said for these attitudes but the difficulty is rather more than Rooth's worry that there is no single right theory but lots of different ones, or even Gawron and Peters' worry that although there is a single right theory we cannot decide what it is. The difficulty seems to stem from the fact that our project is to find linguistic order in the unruly data and show how systematically

recurring features determine sentence meanings. In the data under discussion, however, the order that we discern in other *simpler* sentences just does not seem to project neatly onto our more complex cases.

At least part of the difficulty is no doubt arising from pragmatic influences. For example, our translation of the indefinite determiner 'a' by the existential quantifier is often motivated in part by the observation that 'A man came in' is still *true* when several men came in, even though a helpful and co-operative speaker would not actually use that sentence if the quantity of men that came in was a topic of interest to the hearer. Given this observation, it is perhaps not so surprising that we experience difficulty in evaluating some sentences containing an indefinite noun phrase when precisely the quantity of objects involved *is* the central topic of interest.

Approaches invoking pragmatic influences have been presented of course (for recent examples see [Heim 90] and [Chierchia 91]) but even with these resources it has proved difficult to account for all the variety of data. Another approach suggested by Cooper recently is to relativise quantification to 'collections of situations' whose differing internal structures can give rise either to 'conservative' readings (as in 78b) or to the readings favoured by Kamp and Reyle. What the structure of such a collection of situations is in any given case is not to be determined (wholly) by language but will include pragmatic influences ([Cooper 91]). Whilst these sorts of approaches look promising, I suggest that, in the meantime, $\mathrm{DPL}_{\mathrm{bq}}$ does indeed meet its theoretical aims without making *completely* implausible empirical predictions.

6.6 Conclusions

What has been demonstrated here is that the notion of *threading* information between formulae, which is central to Dynamic Logic, can be extended to logics containing binary quantificational structures, without either encountering the Proportion Problem or making use of other operations such as copying (either Discourse Representation

Structures or whole evaluations of formulae). Dynamic Predicate Logic with Binary Quantifiers (DPL_{bq}) not only provides plausible truth conditions for its formulae (and hence the natural language sentences they represent) but it does so in a reasonably transparent manner. The role of threading and context-change is clearly shown in the rules for the input and output functions and those rules are neatly separated from the rules giving truth conditional content in a more traditional fashion. The latter rules reveal their content by using conjunction (as ordinarily understood) to characterize object language conjunction, implication (as ordinarily understood) to characterize object language implication and binary structured quantification (as ordinarily understood) to characterize object language binary quantification, and so on. Free variables are treated uniformly (within and without the sentence) by use of just one rule. Since this rule appeals directly to possible values for variables made available by the input context for the current formula, and since input contexts are generated by the earlier evaluation of other formulae, DPL_{bq} is a clear example of a dynamic logic incorporating the idea of formulae both being evaluated in a context and themselves altering the context of evaluation for other formulae.

Chapter 7

Conclusions and Future Directions

This thesis has been concerned with the idea of the dynamic interpretation of natural language. At a highly abstract level, this is simply the idea that not only does context influence the interpretation of language but also that the interpretation of language affects the context of interpretation. In particular, the thesis has tackled two main problems.

The first concerned dynamically interpreting English surface structures where decisions about quantifier scope ambiguities and pronominal resolution were not already represented. The resulting theory was able to handle simple cases where the possibility of anaphoric resolution depended on the result of a prior quantifier scoping decision. I presented first a new account of quantifier scope ambiguities which predicted that syntactically unambiguous surface structures containing more than one quantified noun phrase were ambiguous. This was achieved by simply using conditionals rather than biconditionals in the truth theory (together with a general exclusion clause). This method allows multiple routes to be taken through the truth theory for the same input syntactic structure. The relational nature of meaning for ambiguous sentences therefore arises in the semantics and not in the syntax. The account was compared with other extant accounts of quantifier scope ambiguities. The theory was also shown to bear close relation to a quantifier scoping algorithm of Hobbs and Shieber and in fact

some simple improvements to that algorithm were suggested. Two programs based on these ideas appear in the appendices (Appendices A.1, A.2). The account was then extended to include dynamic interpretation of anaphora. In particular, the account made direct appeal to the notion of resolving a pronoun to something available in the current context. Pronoun resolution therefore also took place in the semantics and not simply in advance in the syntax. The alternative strategy of dynamically interpreting indexed structures where the indices arise from purely syntactic considerations was argued to be distinctly implausible and, indeed, impossible given cases where pronoun resolution depends on a prior quantifier scoping decision and where quantifier scoping is not itself syntactically marked. A context was taken to be a sequence of chains of anaphoric dependency amongst unique noun phrase indices, or, if you prefer, particular occurrences of noun phrases. Each chain ended in an object which served as the value of all other members of the chain. As each new noun phrase is interpreted, its index is added into a chain of dependency. Syntactic constraints on anaphoric dependency were incorporated using a version of Higginbotham's Linking Theory. An algorithm for the theory was described and a program also appears in Appendix A.3.

At a methodological level, the semantic theory I gave was in the form of a simple one-stage truth theory. The theory made no reference to any intermediate level of representation which might be thought to be of dubious philosophical status. In this way I maintained the ambition of dynamic logic to provide a non-representational theory of discourse. Although the theory did not provide a meaning for each constituent and was not susceptible to a bottom-up interpretation procedure as in Montague's Universal Grammar, I have claimed that the theory is nevertheless compositional in the sense in which we want it to be, namely it is 'learnable and scrutable' to cite [Davidson 72].

The second main problem tackled concerned how one should extend dynamic predicate logic to a dynamic generalized quantifier logic whilst also using the distinctive mechanism of dynamic logic - namely, the threading of contexts between formulae - to handle

anaphoric references between the two arguments to the quantifiers. I argued that one could not incorporate generalized quantifiers and also thread information between the two arguments if contexts were taken to be single assignment functions. I also showed that two recent attempts to include generalized quantifiers in dynamic semantics did not thread information between the two arguments to the quantifier but made use of additional mechanisms such as evaluating material twice in order to secure appropriate results. Finally I presented a system called $\mathrm{DPL}_{\mathrm{bq}}$ which did incorporate generalized quantifiers and which did thread the output from the first argument into the second argument. The empirical predictions of the system were argued to be not implausible, given the fact that it is in general quite difficult to interpret those English sentences whose interpretation ought to provide the distinguishing cases for the theory.

Dynamic logic is still a very recent development in natural language semantics and there is therefore a vast range of possible future work, both theoretically and empirically.

Dynamic Logic has been stimulated in large part more by theoretical and methodological ambitions than with the aim of covering wider ranges of empirical data. Nevertheless, even here there is room for further development and clarification of the nature of dynamic interpretation. It is, I think, an important lesson from chapter 6 of this thesis that we should be clear about the distinctive contribution of dynamic interpretation, namely the threading of contexts between formulae, and also be clear about when phenomena are indeed handled by that mechanism and when other mechanisms, for example double evaluation, are also invoked. This point, I have made whilst examining recent accounts of anaphora in binary quantificational structures in both DRT and DPL.

Another important theoretical topic, often alluded to in dynamic semantics, and one which I have not addressed in this thesis, is that of incremental interpretation. The topic naturally arises in dynamic semantics from the idea that the interpretation of words and sentences depends in part on the context generated by the earlier evaluation

of other words and sentences. Nevertheless, I think incremental interpretation is an issue that should be clearly separated, conceptually, from that of dynamic interpretation. For one thing, one can write a perfectly good dynamic rule for interpreting conjunction, for example, as follows

$[\![(\phi \wedge \psi)]\!]^{i\,o}$ iff $\exists k\, ([\![\psi]\!]^{i\,k} \wedge [\![\phi]\!]^{k\,o})$

The interpretation of ψ precedes that of ϕ even though ψ follows ϕ in linear order in the formula. Everything that we have said concerning context dependency and context generation holds good of this interpretation rule - but the order of interpretation is simply not left to right. Furthermore, the notion of incremental interpretation is clearly a psychological notion consisting of the idea that what we understand about what is being said is built up, though not necessarily monotonically, as utterances are produced and sentences are read. The more we hear and read of a sentence, the more of what is being said we understand. In this sense, of course, the contexts of dynamic logic are woefully inadequate since they do not encode anything like a notion of the content of what has been uttered so far. However, even if we restrict our attention to some suitable notion of 'content with respect to anaphoric dependencies' then there is an important point to be made. If the contexts of dynamic logic are to reflect the idea of incremental interpretation, then they must be given a theory-external interpretation. That is, they will not be like the assignment functions of FOPL which have no other status than that conferred on them in virtue of the role they play in generating truth conditions for sentences. This point I made in chapter 4. The notion of context developed in chapter 4 - that of chains of anaphoric dependency amongst the indices of noun phrase utterances - stands a better chance of such an interpretation than those of DPL which use logical form variables. There is, I think, a curious tension in dynamic logic as to whether the notions of context and context change potential are to be construed purely theory-internally or whether they are also intended to be psychologically real. If the latter, then, as I suggested for DRT in chapter 2, one would like to see evidence brought to bear for this interpretation other than that the resulting truth conditions

for the input sentences are intuitively correct.

Incremental interpretation is one of the advantages [Groenendijk & Stokhof 91b] claim for compositional approaches in general. They state that compositionality is a 'most intuitive' way to meet the demand for incremental interpretation, whereas DRT with its two stage process of interpretation requires first the building of a representation for a syntactic unit only after which one can begin interpretation. This is a strange claim since it might easily be thought that compositional interpretation is itself somewhat at odds with incremental interpretation. Compositional interpretation, at least in Groenendijk and Stokhof's eyes, requires meanings for constituent structures whereas incremental interpretation requires an interpretation after each word. It is not immediately obvious how the two are to be reconciled though the two are not in fact incompatible. One possibility is to provide suitably right branching tree-structures for sentences in the style of [Ades & Steedman 82]. Another, and I think more attractive, possibility is to resist the temptation to identify the result of incremental interpretation with that of the denotation of the (structure of the) words produced so far. In [Pulman 85], for example, the result of incremental interpretation is a state which includes expectations about what will be encountered. The interpretation associated with a state after a sentence initial 'the farmer' is $\lambda vp.vp(the'(farmer'))$, even though the interpretation of the phrase 'the farmer' is itself simply $the'(farmer')$. In general, denotations simply tell one what a phrase will contribute to the truth conditions of a sentence in which it appears. So the rule that $[\![John]\!] = John$ tells us (in combination with the other axioms of the theory) that the interpretation of any sentence containing 'John' will turn on how things are with John. So a hearer who encounters 'John' will know that the truth of the current utterance depends somehow on John but knowledge of this fact is not itself part of the denotation, which is just John himself. So we can distinguish denotations of constituent structures from the results of incremental interpretation.

One possible reason why Groenendijk and Stokhof associate compositionality with incrementality is that by using DPL one can write down the translation of each sentence

as it occurs. Given an initial sentence 'A man walks', one can write down its translation $\exists x\ (m(x) \wedge w(x))$ and given a subsequent 'He whistles' one can just write down $wh(x)$. The theory therefore appears to be incremental at the level of the sentence. Actually, I think some caution should be attached to such a claim. First, DRT, which is held to be non-compositional, is also suitably incremental at the level of the sentence in the sense that each sentence is treated one by one and after each has been processed the resulting DRS can be evaluated for truth or falsity. Secondly, when the compositional system is extended 'below the level of the sentence' in Dynamic Montague Grammar, the result is patently not incremental - for DMG again generates formulae and meanings for constituent structures and not for initial substrings. Finally, there are sentences such as 'A man walks and he whistles' where an explicit conjunction is used. These sentences are not treated incrementally by DPL since there is no formula or meaning for 'A man walks and' - yet it is difficult to see quite why this sentence should differ so significantly from the discourse 'A man walks. He whistles'. If in the latter case we can say that 'sentence concatenation signifies conjunction' why cannot explicit use of a suitable lexical connective perform the same task?

Dynamic interpretation, incremental interpretation and compositional interpretation are three separate, if related, notions. I think that the proper relation between the states required by dynamic interpretation and those of incremental interpretation may prove to be one of the most important theoretical issues confronting the development of dynamic logic.

On the empirical side, prominent areas closely related to the sorts of case considered here include one-anaphora, vp-anaphora (for which see [Gardent 91] for one possible approach), *wh*-phenomena, questions and plurals. From the perspective of this thesis, an approach to plurals building on the use of Linking Theory as in chapter 4 looks most promising. Linking Theory was originally developed with certain plural anaphora in mind, notably cases of 'split antecedents' such as 'John told Mary that they should leave'. In these cases, one cannot take the value of the pronoun to be simply that

of its antecedent since the pronoun has two antecedents. Much recent work in DRT has been devoted to cases such as these ([Kamp & Reyle *forth.*]) with various additions to the rules which construct and modify DRSs being proposed to handle them. For example, there are rules for joining discourse referents together to form complex objects and also rules for abstracting and summing over discourse referents within a nested DRS. The results themselves form a new type of discourse referent which plural pronouns can then refer to. Some general principles governing such rules are proposed - for example, that the rules may only construct antecedents for pronouns by collecting together discourse referents already introduced into the DRS. No 'subtraction' of entities is allowed. These principles are interpreted as empirical generalizations about the deductions and inferences that humans actually make whilst in the process of interpreting sentences. It is an interesting question to what extent it is possible - or even desirable - to construct a dynamic logic with a similar coverage but without all or even some of these psychological pretensions. For the cases of split antecedents, it does not look implausible to claim that the value of a pronoun will be the construction of the values of all the antecedents to which it is linked - though there appears to be at least no necessity to interpret this rule as representing an actual inference that anyone makes.

Finally, with respect to the computation of logical forms for sentences, dynamic logic offers interesting new possibilities for systems which aim to integrate contextual and context-independent effects on meanings. For example, computational linguists have often stressed the importance in their theories of the building and exploiting of discourse models whilst interpreting language. Dynamic Logic holds out the promise of a formal semantical analysis of such theories. It also offers some new capabilities. For example, I have claimed in chapter 5 that even existing systems with discourse models found themselves resorting to rather ad-hoc rules to handle intra-sentential donkey anaphora - yet recent theories in discourse semantics have aimed and succeeded in supplying a uniform account of certain intra-sentential and inter-sentential anaphora.

The algorithm I presented dealt with them both in the same fashion. Dynamic logic can also be taken as offering a new meaning representation formalism. That is, one could take as one's objective the production of meaning representations themselves to be understood dynamically. The development of proof theories for dynamic logics make this an intriguing possibility. In this thesis, I have taken the alternative line of producing static meaning representations in first order logic and keeping the dynamic element in the interpretation process itself. In this way, we can both use dynamic mechanisms and help ourselves to the large amount of computational work on first order logic which can be applied to the output representations. Which strategy will prove the better in the long term is an open question.

Dynamic Logic represents an interesting and important new building material for the semantic mason. The investigation of its peculiar binding properties gives us the prospect of an analysis of new and significant structures in the overall architecture of our language.

Bibliography

[Ades & Steedman 82] A Ades and M. Steedman. On the order of words. *Linguistics and Philosophy*, 4:517–558, 1982.

[Alshawi & Crouch 92] H. Alshawi and R. Crouch. Monotonic semantic interpretation. Paper to be presented at ACL Newark, Delaware, 1992.

[Alshawi 90] H. Alshawi. Resolving quasi logical forms. *Computational Linguistics*, 16 number 3:133–144, 1990.

[Alshawi 92] H. Alshawi. *The Core Language Engine*. M.I.T.Press, 1992.

[Alshawi *et al* 88] H. Alshawi, D.M. Carter, J. van Eijck, R.C. Moore, D.B. Moran, F.C.N. Pereira, A.G. Smith, and S.G. Pulman. *Interim Report on the SRI Core Language Engine*. Cambridge Computer Science Research Centre, 1988. Report CCSRC-005.

[Barwise & Cooper 82] J. Barwise and R. Cooper. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219, 1982.

[Barwise 87] J. Barwise. Noun phrases, generalized quantifiers and anaphora. In P. Gärdenfors, editor, *Generalized Quantifiers*, pages 1–29. Reidel, 1987.

[Chierchia 88] G. Chierchia. Dynamic generalized quantifiers and donkey anaphora. In M. Krifka, editor, *Proceedings of the 1988 Tubingen Conference on Genericity in Natural Language*, pages 53–84. University of Tubingen, 1988.

[Chierchia 91] G. Chierchia. Anaphora and dynamic logic. In M. Stokhof, J. Groenendijk, and Beaver D., editors, *Quantification and Anaphora 1*. DYANA Deliverable R2.2A, 1991.

[Chierchia 92] G. Chierchia. Anaphora and dynamic binding. *Linguistics and Philosophy*, 15:111–183, 1992.

[Chomsky 81] N. Chomsky. *Lectures on Government and Binding.* Foris, 1981.

[Cooper 79] R. Cooper. The interpretation of pronouns. In F. Heny and H.S. Schnelle, editors, *Syntax and Semantics*, volume 10, pages 61–92. Academic Press, 1979.

[Cooper 83] R. Cooper. *Quantification and Syntactic Theory.* Studies in Linguistics and Philosophy, 21. Reidel, 1983.

[Cooper 91] R. Cooper. Donkeys and situations. In M. Stokhof, J. Groenendijk, and Beaver D., editors, *Quantification and Anaphora 1.* DYANA Deliverable R2.2A, 1991.

[Davidson 68] D. Davidson. On saying that. *Synthese*, 19:158–174, 1968. Also in his *Inquiries into Truth and Interpretation*, Oxford, 1984.

[Davidson 72] D. Davidson. Semantics for natural languages. In D. Davidson and G. Harman, editors, *Semantics of Natural Language.* Reidel, 1972. Also in his *Inquiries into Truth and Interpretation*, Oxford, 1984.

[Dowty *et al* 81] D. Dowty, R. Wall, and S. Peters. *Introduction to Montague Semantics.* Studies in Linguistics and Philosophy, 11. Reidel, Dordrecht, 1981.

[Emms 90] M. Emms. Polymorphic quantifiers. In G. Barry and G. Morrill, editors, *Studies in Categorial Grammar.* Edinburgh Working Papers in Cognitive Science, vol. 5, 1990.

[Evans 77] G. Evans. Pronouns, quantifiers and relative clauses i. *Canadian Journal of Philosophy*, 7:467–536, 1977.

[Evans 80] G. Evans. Pronouns. *Linguistic Inquiry*, 11(2):337–362, 1980.

[Evans 85] G. Evans. Does tense logic rest on a mistake ? In *Collected Papers.* Oxford, 1985.

[Gardent 91] C. Gardent. Vp anaphora. Unpublished PhD thesis, 1991.

[Gawron & Peters 90] M. Gawron and S. Peters. Some puzzles about pronouns. In R. Cooper, K. Mukai, and J. Perry, editors, *Situation Theory and its Applications, I.* Chicago University Press, 1990.

[Groenendijk & Stokhof 91a] J. Groenendijk and M. Stokhof. Dynamic montague grammar. In J. van Bentham, editor, *Quantification and Anaphora 1.* DYANA Deliverable R2.2A, 1991.

[Groenendijk & Stokhof 91b] J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–100, 1991.

[Grosz *et al* 86] B.J. Grosz, K. Sparck Jones, and B.L. Webber. *Readings in Natural Language Processing.* Morgan Kaufman, California, 1986.

[Harel 84] D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic.* Reidel, 1984.

[Heim 82] I. Heim. *Definite and Indefinite Noun Phrases.* Unpublished PhD thesis, University of Massachusetts, Amherst, 1982.

[Heim 90] I. Heim. E-type pronouns and donkey anaphora. *Linguistics and Philosophy*, 13:137–177, 1990.

[Hendriks 90] H. Hendriks. Flexible montague grammar. In H. Hendriks and M. Moortgat, editors, *Theory of Flexible Interpretation: Specification and Grammar Fragment.* DYANA Deliverable R1.2A, 1990.

[Higginbotham 83] J. Higginbotham. Logical form, binding and nominals. *Linguistics and Philosophy*, 6:395–420, 1983.

[Higginbotham 85] J. Higginbotham. On semantics. *Linguistic Inquiry*, 16(4), 1985.

[Hintikka 74] J. Hintikka. Quantifiers vs. quantification theory. *Linguistic Inquiry*, 5:153–177, 1974.

[Hobbs & Shieber 87] J.R. Hobbs and S.M. Shieber. An algorithm for generating quantifier scopings. *Computational Linguistics*, 13 numbers 1-2:47–63, 1987.

[Isard 75] S.D. Isard. Changing the context. In E. Keenan, editor, *Formal Semantics of Natural Language.* Cambridge University Press, 1975.

[Janssen 83] T.M.V. Janssen. *Foundations and Applications of Montague Grammar.* Mathematisch Centrum, Amsterdam, 1983.

[Kamp & Reyle *forth.*] H. Kamp and U. Reyle. *From discourse to logic: Introduction to Model Theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory.* Kluwer. Forthcoming.

[Kamp 81] H. Kamp. A theory of truth and semantic representation. In J. Groenendijk, T. Janssen, and M. Stokhof, editors, *Formal Methods in the Study of Language.* Mathematical Center, Amsterdam, 1981.

[Kamp 90] H. Kamp. Comments on dynamic predicate logic. In J. van bentham, editor, *Partial and Dynamic Semantics I.* DYANA Deliverable R2.1A, 1990.

[Kaplan 89] D. Kaplan. Demonstratives. In J. Almog, D. Kaplan, J. Perry, and H.K. Wettstein, editors, *Themes from Kaplan*, pages 481–563. Oxford, 1989. Published version of long unpublished 1977 manuscript 'Demonstratives' Draft 2 UCLA.

[Keller 86] W. Keller. Nested cooper storage. In U. Reyle and C. Rohrer, editors, *Natural Language Parsing and Linguistic Theory*, Studies in Linguistics and Philosophy volume 35, pages 432–447. Reidel, 1986.

[Landman 83] I. Landman, F. & Moerdijk. Compositionality and the analysis of anaphora. *Linguistics and Philosophy*, 6:89–114, 1983.

[Lasnik 76] H. Lasnik. Remarks on coreference. *Linguistic Analysis*, 2:1–22, 1976.

[Latecki & Pinkal 90] L. Latecki and M. Pinkal. Syntactic and semantic conditions for quantifier scope. Technical Report Arbeitspapier Nr. 13, Universität des Saarlandes, Germany, 1990. also in: 'Proceedings of Workshop on Processing of Plurals and Quantifiers' GWAI-90.

[Lewin 90] I. Lewin. A quantifier scoping algorithm without a free variable constraint. In *Proceedings of the 13th International Conference on Computational Linguistics, Vol. 3*, pages 190–194, Helsinki, Finland, 1990.

[Lewin 91] I. Lewin. Threading in binary quantificational structures. In J. van der Does, editor, *Quantification and Anaphora*, volume 2. DYANA Deliverable, 1991. R2.2B.

[Lewis 81] D.K. Lewis. Index, context and content. In S. Kanger and S. Ohman, editors, *Philosophy and Grammar.* Reidel, 1981.

[Matilal & Sen 90] B.K. Matilal and P.K. Sen. The context principle and some indian controversies over meaning. *Mind*, 1990.

[May 88] R. May. Bound variable anaphora. In R.M. Kempson, editor, *Mental Representations: the interface between language and reality*, pages 85–104. Cambridge Univeristy Press, 1988.

[Montague 74a] R. Montague. The proper treatment of quantification in English. In Thomason R., editor, *Formal Philosophy.* Yale University Press, New York, 1974.

[Montague 74b] R. Montague. Universal grammar. In Thomason R., editor, *Formal Philosophy*. Yale University Press, New York, 1974.

[Partee & Bach 80] B. Partee and E. Bach. Anaphora and semantic structure. In *Papers from the Parasession on Pronouns and Anaphora*, volume 10. Chicago Linguistic Society, 1980.

[Partee & Bach 81] B. Partee and E. Bach. Quantification, pronouns and vp anaphora. In J. Groenendijk, T. Janssen, and M. Stokhof, editors, *Formal Methods in the Study of Language*, pages 445–481. Mathematical Center, Amsterdam, 1981.

[Partee 76] B.H. Partee. Some transformational extensions of montague grammar. In B.H. Partee, editor, *Montague Grammar*. Academic Press, 1976.

[Partee 79] B.H. Partee. Montague grammar and the well-formedness constraint. In F. Heny and S. Schnelle, editors, *Syntax and Semantics 10*, pages 275–313. Academic Press, 1979.

[Partee 83] B. Partee. Nominal and temporal anaphora. *Linguistics and Philosophy*, 7:243–286, 1983.

[Partee 84] B. Partee. Compositionality. In F. Landman and F. Veltman, editors, *Varieties of formal semantics: Proceedings of the Fourth Amsterdam Colloquium*, pages 281–311. Foris, 1984.

[Pereira & Pollack 91] F.C.N. Pereira and M.E. Pollack. Incremental interpretation. *Artificial Intelligence*, 50:37–82, 1991.

[Pereira 90] F.C.N. Pereira. Categorial semantics and scoping. *Computational Linguistics*, 16:1–9, 1990.

[Pulman 85] S.G. Pulman. A parser that doesn't. In *Proceedings of the 2nd European ACL, Geneva*, pages 128–135, 1985.

[Reinhart 83] T. Reinhart. Coreference and bound anaphora: A restatement of the anaphora questions. *Linguistics and Philosophy*, 6:47–88, 1983.

[Rescher 62] N. Rescher. Plurality quantification. *Journal of Symbolic Logic*, 27:373–374, 1962.

[Rooth 87] M. Rooth. Noun phrase interpretation in montague grammar, file change semantics and situation semantics. In P. Gärdenfors, editor, *Generalized Quantifiers*, pages 237–268. Reidel, 1987.

[Schubert & Pelletier 82] L.K. Schubert and F.J. Pelletier. From english to logic: Context free computation of "conventional" logical translation. *American Journal of Computational Linguistics*, 8:27–44, 1982.

[Schubert & Pelletier 89] L.K. Schubert and F.J. Pelletier. Generically speaking, or, using discourse representation theory to interpret generics. In G. Chierchia, B.H. Partee, and R. Turner, editors, *Properties, Types and Meaning II*, pages 193–268. Kluwer Academic, 1989.

[Strawson 70] P.F. Strawson. Categories. In O.P. Wood and G. Pitcher, editors, *Ryle*. London Macmillan, 1970.

[Tarski 56] A. Tarski. The concept of truth in formalised languages. In *Logic,Semantics, Metamathematics*. Oxford University Press, 1956.

[van Bentham 86] J. van Bentham. *Essays in Logical Semantics*. Studies in Linguistics and Philosophy 29. D.Reidel, 1986.

[van Bentham 91] J. van Bentham. *Language in Action - Categories, Lambdas and Dynamic Logic*, volume 130 of *Studies in Logic and the Foundations of Mathematics*. North Holland, 1991.

[van Eijck & de Vries 91] J. van Eijck and F.J. de Vries. Dynamic interpretation and hoare deduction. Technical Report CS-R9115, Centre for Mathematics and Computer Science, Amsterdam, Holland, 1991. also to appear in Journal of Logic, Language and Information.

[van Eijck 85] J. van Eijck. *Aspects of Quantification in Natural Language*. Unpublished PhD thesis, Rijksuniversiteit te Groningen, Groningen, Holland, 1985.

[Woods 68] W. Woods. Procedural semantics for a question-answering machine. In *Proceedings of the Fall Joint Computer Conference, New York*, pages 457–471. New York, 1968.

[Woods 78] W. Woods. Semantics and quantification in natural language question answering. In Yovits, editor, *Advances in Computers*, pages 2–64. Academic Press, New York, 1978.

[Zeevat 89] H. Zeevat. A compositional approach to discourse representation theory. *Linguistics and Philosophy*, 12:95–131, 1989.

Appendix A

Programs

A.1 An HSL Program

```
% Program: A quantifier scoping algorithm for the language HSL

% A wff of form p(arg1...argn) is represented by wff(p,[arg1'...argn'])
% where arg1' is the encoding of arg1
%
% A complex term <q v r> is represented by term(q,v,r') where r' encodes r

% Example:
%    wff(see,[jack,term(every,x,wff(man,[x]))]) encodes
%    see(jack,<every,x,man(x)>)
% Example:
%    some rep. of every dept in most companies saw a few samples of each product
%    (Hobbs and Shieber's 42 reading sentence:
%    wff(see,[term(some,
%                  r,
%                  wff(and,[wff(rep,[r]),
%                           wff(of,[r,term(every,
%                                          d,
%                                          wff(and,[wff(dept,[d]),
%                                                   wff(in,[d,term(most,
%                                                                  c,
%                                                                  wff(com,[c]))])])
%                                          )
%                                   ]
%                              )
%                           ]
%                     )
%                  ),
%             term(afew,
%                  s,
%                  wff(and,[wff(samp,[s]),
%                           wff(of,[s,term(each,
%                                          p,
%                                          wff(prod,[p]))])])))
```

```
%            ]
%       ).
%
```

```
% gen(Form,Scopedform)
% ====================
% Form       => An input wff to be scoped
% Scopedform <= A scoped version of Form
%
% True if 2) is a scoped version of 1)

gen(Form,Scopedform)   :-   apply_terms(Form,Scopedform).

% apply-terms(Form,Newform)
% =========================
% Form    => a formula in the input language
% Newform <= a formula in the output language
%
% True if Newform is the result of applying our
% translation rules to Form
%
% a) find an applicable term in Form
% b) substitute its variable for that term in Form
% c) recurse (applying all the other applicable terms) on the result of b)
% d) recurse on the restriction
% d) build the final result

apply_terms(Form,Form)
  :- \+ applicable_term(Form,_).

apply_terms(Form,Scopedform)
  :-
     applicable_term(Form,term(Quant,Var,Restrict)),
     subst(Var,term(Quant,Var,Restrict),Form,Lesser_form),
     apply_terms(Lesser_form,New_Lesser),
     apply_terms(Restrict,Newrestrict),
     Scopedform = wff(Quant,Var,Newrestrict,New_Lesser).


% applicable_term(Form,Term)
% ==========================
% Form => A wff
% Term <= A complex term contained in Form
%
% True if Term is an applicable term contained in Form

applicable_term(wff(_Pred,Args),Term)
  :-
     applicable_term(Args,Term).

applicable_term([],[])
```

```
  :- !,fail.

applicable_term([H|T],Term)
  :-
     (applicable_term(H,Term)
      ;
      applicable_term(T,Term)
     ).

applicable_term(Term,Term) :- complex_term(Term).

applicable_term(term(_,_,Restrict),Term)
  :-
     applicable_term(Restrict,Term).

%=============================================================
% Low level calls

% subst(Var,Term,Body,Outbody)
% ============================
% Var       => variable to be substituted in
% Term      => term to be replaced by Var
% Body      => input formula to be altered into ..
% Outbody   <= output formula
%
% True if Outbody is the same as Body, but with all occurrences
% of Term replaced by Var.

% no more Body to look at
subst(_,_,[],[]) :- !.

% The body is a wff - look at the args
subst(Var,
      Term,
      wff(Pred,Args),
      wff(Pred,Newargs))
  :-
      !,
      subst(Var,Term,Args,Newargs).

% The body is a list - recurse down it
subst(Var,
      Term,
      [H_arg|T_args],
      [New_H_arg|New_T_args])
  :-
      !,
      subst(Var,Term,H_arg,New_H_arg),
      !,
      subst(Var,Term,T_args,New_T_args).

% The body is what we're looking for - replace with var.
subst(Var, Term, Term, Var) :- !.
```

```
% The body is a complex term - but not the one we want
subst(Var,
      Term,
      term(Quant,NVar,Restrict),
      term(Quant,NVar,Newrestrict))
  :-
      !,subst(Var,Term,Restrict,Newrestrict).

subst(_,_,Body,Body).

complex_term(term(_,_,_)).
```

A.2 An HSL+ Program

```
% Program: A quantifier scoping algorithm for the language HSL+

% A wff of form p(arg1...argn) is represented by wff(p,[arg1'...argn'])
% where arg1' is the encoding of arg1
%
% A complex term <q v r> is represented by term(q,v,r') where r' encodes r

% Example:
%    wff(not,[wff(here,[term(every,x,wff(man,[x]))])]) encodes
%    not(here,<every,x,man(x)>)

% gen(Form,Scopedform,Pol)
% ========================
% Form       => An input wff to be scoped
% Scopedform <= A scoped version of Form
% Polarity   => 1 or 0 (looking for truth or falsity conditions)
%
% True if 2) is a scoped version of 1)

gen(Form,Scopedform,Pol)   :-   apply_terms(Form,Scopedform,Pol).

% apply-terms(Form,Newform,Pol)
% =============================
% Form    => a formula in the input language
% Newform <= a formula in the output language
% Polarity   => 1 or 0 (looking for truth or falsity conditions)
%
% True if Newform is the result of applying our
% translation rules to Form
%
% a) find an applicable term in Form
% b) substitute its variable for that term in Form
% c) recurse (applying all the other applicable terms) on the result of b)
% d) recurse on the restriction
% d) build the final result

apply_terms(Form,Scopedform,Pol)
  :-
     applicable_term(Form,term(Quant,Var,Restrict)),
     subst(Var,term(Quant,Var,Restrict),Form,Lesser_form),
     apply_terms(Lesser_form,New_Lesser,Pol),
     apply_terms(Restrict,Newrestrict,1),
     dual(Quant,Pol,NewQuant),
     Scopedform = wff(NewQuant,Var,Newrestrict,New_Lesser).

apply_terms(wff(not,Form),Scopedform,Pol)
  :-
     dual(Pol,NewPol),
     apply_terms(Form,Scopedform,NewPol).

apply_terms(wff(Conj,[A,B]),Scopedform,Pol)
  :- dual(Conj,Pol,Newconj),
```

```
      apply_terms(A,ScopedA,Pol),
      apply_terms(B,ScopedB,Pol),
      Scopedform = wff(Newconj,[ScopedA,ScopedB]).

apply_terms(Form,Scopedform,Pol)
  :- \+ applicable_term(Form,term(_,_,_)),
      \+ (Form = wff(Pred,_),member(Pred,[not,and,or])),
      base_case(Form,Scopedform,Pol).

base_case(Form,Form,1).
base_case(Form,wff(not,Form),0).


% applicable_term(Form,Term)
% ==========================
% Form => A wff
% Term <= A complex term contained in Form
%
% True if Term is an applicable term contained in Form

% Form is an ordinary wff - look down the list of Args

applicable_term(wff(Pred,_Args),_Term)
  :- member(Pred,[and,or]),
      !,fail.                              % don't look in a conjunction

applicable_term(wff(_Pred,Args),Term)
  :-
      applicable_term(Args,Term).

% Form is a list of Args - recurse down it

applicable_term([],[])
  :- !,fail.

applicable_term([H|T],Term)
  :-
      (applicable_term(H,Term)
       ;
       applicable_term(T,Term)
      ).

applicable_term(Term,Term) :- complex_term(Term).

% Form is a complex term - look inside it

applicable_term(term(_,_,Restrict),Term)
  :-
      applicable_term(Restrict,Term).

%=============================================================
%=============================================================
% Low level calls
```

```
% subst(Var,Term,Body,Outbody)
% ============================
% Var       => variable to be substituted in
% Term      => term to be replaced by Var
% Body      => input formula to be altered into ..
% Outbody   <= output formula
%
% True if Outbody is the same as Body, but with all occurrences
% of Term replaced by Var.

% no more Body to look at
subst(_,_,[],[]) :- !.

% The body is a wff - look at the args
subst(Var,
      Term,
      wff(Pred,Args),
      wff(Pred,Newargs))
  :-
      !,
      subst(Var,Term,Args,Newargs).


% The body is a list - recurse down it
subst(Var,
      Term,
      [H_arg|T_args],
      [New_H_arg|New_T_args])
  :-
      !,
      subst(Var,Term,H_arg,New_H_arg),
      !,
      subst(Var,Term,T_args,New_T_args).

% The body is what we're looking for - replace with var.
subst(Var, Term, Term, Var) :- !.

% The body is a complex term - but not the one we want
subst(Var,
      Term,
      term(Quant,NVar,Restrict),
      term(Quant,NVar,Newrestrict))
  :-
      !,
      subst(Var,Term,Restrict,Newrestrict).

% The body is a different atomic case - leave as is.
subst(_,_,Body,Body).

complex_term(term(_,_,_)).

dual(X,1,X).
dual(and,0,or).
dual(or,0,and).
```

```
dual(all,0,a).
dual(a,0,all).
dual(every,0,some).
dual(some,0,every).
dual(1,0).
dual(0,1).


member(X,[X|_]).
member(X,[_|T]) :- member(X,T).
```

A.3 An RL+ Program

A.3.1 A Rooth Fragment Parser

```
% A DCG to parse the Rooth Grammar plus build an H&S input rep.
% ------------------------------------------------------------
% Rules

d(LF)              --> sent(LF).
d(wff(and,[S,D])) --> sent(S),d(D).

sent(S)                --> s([Indx],S),{gnum(Indx)}.
sent(wff(and,[S1,S2])) --> s([Indx],S1),{gnum(Indx)},
                           [and],
                           sent(S2).
sent(wff(imp,[S1,S2])) --> [if],
                           s([Indx],S1),{gnum(Indx)},
                           sent(S2).


s(Indx,wff(Pred,[NP|Args])) --> np(Indx,NP),
                                vp(Indx,wff(Pred,Args)).
vp(Ind,wff(V,[NP]))         --> v(V),
                                np([Y|Ind],NP), {gnum(Y)}.

np(Indx,term(DET,Var,wff(and,[wff(N,[Var]),wff(Prep,[Var,NP])])))
                            --> det(DET),
                                n(N),{genvar(Var)},
                                pp(Indx,wff(Prep,[Var,NP])).

np(_,term(Det,Var,wff(N,[Var])))
                            --> det(Det),
                                n(N), {genvar(Var)}.

pp(Indx,wff(Prep,[_,NP]))   --> p(Prep),
                                np([X|Indx],NP), {gnum(X)}.


% Lexicon

vp(_Ind,wff(X,[]))      --> [X], {vps(X)}.
v(X)                    --> [X], {vs(X)}.
np(_,pt(Uniq,X))        --> [X], {pts(X),genvar(Uniq)}.
np(_,pn(Uniq,X))        --> [X], {pns(X),genvar(Uniq)}.
det(X)                  --> [X], {dets(X)}.
n(X)                    --> [X], {ns(X)}.
p(X)                    --> [X], {ps(X)}.

vps(X)  :- member(X,[walked,satdown,ran,arrived,swam]).
vs(X)   :- member(X,[loved,beat,saw,kicked]).
pts(X)  :- member(X,[he,him,she,her,it,himself,herself,itself]).
pns(X)  :- member(X,[john,mary,bill,sue,jack]).
dets(X) :- member(X,[every,some,a,most]).
```

```
ns(X)    :- member(X,[priest,man,dog,pig,woman,donkey,boy]).
ps(X)    :- member(X,[with]).
```

A.3.2 A Scoping and Resolution Program

```
:- op(300,yfx,-).
:- op(401,yfx,&).

% gen(Form,Scopedform)
% ====================
% Form       => An input wff to be scoped
% Scopedform <= A scoped version of Form
%
% True if 2) is a scoped and resolved version of 1)
%
% * find all proper name in discourse and add them into
%   the initial discourse context of []: treats them as discourse
%    wide quantifiers
% * Scope with amended input context

gen(Form,Scopedform)
 :- baggof([B,C],applicable_term(Form,pn(B,C)),Listofnames),
    append(Listofnames,[],Nvar),
    pull(Form,Scopedform,Nvar).

% pull(Form,Scopedform,Ivar)
% ==========================
% True if 2) is a scoped and resolved version of 1) in input context Ivar
%
% We process Form in Ivar generating a output form (Body) and an output
% context (Ovar). Then we existentially quantify over variables that occur
% in Ovar and not in Ivar

pull(Form,Scopedform,Ivar)
 :-
    pull(Form,Body,Ivar,Ovar),
    smp(Ivar,Ovar,Body,Scopedform).

% smp(Ovar,Dom,Body,Scopedform)
% =============================
% Ovar is the input context range and Dom is
% the output context range - all those objects
% in the output that weren't in the input
% are to be quantified over at this level
% (if they are in the input then they're governed
% by a higher quantifier)

smp(Ivar,Ovar,Body,Scopedform)
 :- diff(Ivar,Ovar,Diff),            % find the diff between input and output
    result(Diff,Body,Scopedform).    % quantify over the diff.

diff(Ivar,Ovar,Diff)
 :- collapse(Ivar,Inew),
    collapse(Ovar,Onew),
    differ(Inew,Onew,Diff).
```

```
collapse([],[]).
collapse([H|T],[F|Result])
 :- last(F,H),
    collapse(T,Result).

differ(_Inew,[],[]).
differ(Inew,[H|T],[H|R])
 :- \+ member(H,Inew),!,
    differ(Inew,T,R).
differ(Inew,[_H|T],R)
 :- differ(Inew,T,R).

result([],Body,Body)  :- !.
result(Diff,Body,exists(Diff,Body)).

% pull(Form,Scopedform,I,O)
% =========================
% Form       => An input wff to be scoped
% Scopedform <= A scoped version of Form
% I          => Input context
% O          <= Output context
%
% True if Scopedform is a translation of Form, by virtue of
% applying one of the rules (terms,and,imp,atomic)

pull(Form,Scopedform,I,O)  :-   apply_terms(Form,Scopedform,I,O).

pull(Form,Scopedform,I,O)  :-   apply_and(Form,Scopedform,I,O).

pull(Form,Scopedform,I,O)  :-   apply_imp(Form,Scopedform,I,O).

pull(Form,Scopedform,I,O)  :-   apply_atomic(Form,Scopedform,I,O).


% apply-terms(Form,Scopedform,I,O)
% ================================
%
% a) find an applicable term in Form
% b) substitute its variable for that term in Form - call this 'Body'
% c) recurse on Restrict*Body (* = 'and' for \exists and '->' for \forall
% d) build the final result

apply_terms(Form,Scopedform,Ivar,O)
  :-
     applicable_term(Form,term(Quant,Var,Restrict)),
     apply_qterm(Form,term(Quant,Var,Restrict),Scopedform,Ivar,O).

apply_qterm(Form,term(a,Var,Restrict),Scopedform,Ivar,O)
  :- !,
     subst(Var,term(a,Var,Restrict),Form,Lesser_form),
     pull(wff(and,[Restrict,Lesser_form]),Scopedform,[[Var]|Ivar],O).

apply_qterm(Form,term(Quant,Var,Restrict),Scopedform,Ivar,Ivar)
  :-
```

```
      subst(Var,term(Quant,Var,Restrict),Form,Lesser_form),
      pull(wff(imp,[Restrict,Lesser_form]),PartScopedform,[[Var]|Ivar]),
      Scopedform = every([Var],PartScopedform).


% apply_and(Form,Scopedform,I,O)
% ==============================
%
% the obvious clause for "and"

apply_and(wff(and,[S1,S2]),wff(and,[News1,News2]),Ivar,Ovar)
   :-
      pull(S1,News1,Ivar,O),
      pull(S2,News2,O,Ovar).


% apply_imp(Form,Scopedform,I,O)
% ==============================
%
% Similar to 'apply_and' except that
% a) the consequent is evaluated using pull/3 not
%    pull/4 - corresponding to the rule that for
%    every output for the antecedent THERE IS AN
%    output for the consequent
% b) we also universally quantify over the difference
%    between the input and output of the antecedent

apply_imp(wff(imp,[S1,S2]),Scopedform,Ivar,Ivar)
   :-
      pull(S1,News1,Ivar,O),
      pull(S2,News2,O),
      smpa(Ivar,O,News1,News2,Scopedform).

smpa(Ivar,O,News1,News2,Scopedform)
   :- diff(Ivar,O,Diff),
      resulta(Diff,News1,News2,Scopedform).

resulta([],News1,News2,wff(imp,[News1,News2])) :- !.
resulta(Diff,News1,News2,every(Diff,wff(imp,[News1,News2]))).



% apply_atomic(Form,Scopedform)
% =============================
%

apply_atomic(wff(P,Args),wff(P,OArgs),Ivars,Ovars)
    :- \+ terms(wff(P,Args)),
       \+ member(P,[and,imp]),
       replace(Args,OArgs,Ivars,Ovars).


% replace(Args,OArgs,Ivars,Ovars)
% ===============================
```

```
% True if Ovars is the result of adding in the
% indices in Args into the context Ivars. OArgs
% is the like Args except that the members of
% of Args are replaced by their values according
% to Ovars

replace(Args,OArgs,Ivars,Ovars)
 :- baggof(A,nonreflex(A,Args),NonreflList),     % find non-refl pronouns
    addpron(NonreflList,Args,Ivars,Newvars),     % add them into the context
    baggof(B,reflex(B,Args),ReflList),           % find refl pronouns
    addanap(ReflList,Args,Newvars,Ovars),        % add them into the context
    subs(Args,Ovars,OArgs).                      % replace indices by values


nonreflex(pt(Ind,Lex),Args)
 :- member(pt(Ind,Lex),Args),nonrefl(Lex).
reflex(pt(Ind,Lex),Args)
 :- member(pt(Ind,Lex),Args),refl(Lex).

addpron([],_,Ovars,Ovars).
addpron([H|T],Args,Ivars,Newvars)
 :- addin(H,Args,Ivars,Pvars),
    addpron(T,Args,Pvars,Newvars).

% addin(Nonrefl,Args,I,O)
% =======================
% True if O is a new context derived by adding Nonrefl into I.
% Check that no member of the chain is local and that the
% previous entry precedes this one

addin(pt(H,_),Args,[[F|T]|Rest],[[H,F|T]|Rest])
 :- disjoint([F|T],Args),
    precede(F,H).

addin(pt(H,P),Args,[F|B],[F|Ovars])
 :- addin(pt(H,P),Args,B,Ovars).

disjoint([],_).
disjoint([H|T],Args)
 :- \+ anymem(H,Args),
    disjoint(T,Args).

anymem(H,Args) :- member(H,Args).
anymem(H,Args) :- member(pn(H,_),Args).
anymem(H,Args) :- member(pt(H,_),Args).


addanap([],_,Ovars,Ovars).
addanap([H|T],Args,Ivars,Newvars)
 :- addintwo(H,Args,Ivars,Pvars),
    addanap(T,Args,Pvars,Newvars).

% addintwo(Refl,Args,I,O)
% =======================
```

```
% True if O is a new context derived by adding Refl into I.
% Check that there is local preceding antecedent in the chain

addintwo(pt(H,_),Args,[[A|B]|T],[[H,A|B]|T])
 :- member(K,[A|B]),
    anymem(K,Args),
    precede(K,H).

addintwo(pt(H,P),Args,[F|B],[F|Ovars])
 :- addintwo(pt(H,P),Args,B,Ovars).


% subs(Args,Ovars,OArgs)
% ======================
% replace each member of Args by its value according to Ovars giving OArgs

subs([],_,[]).
subs([pt(H,_Lex)|T],Oargs,[F|R])
 :- !,member(K,Oargs),
    member(H,K),
    last(F,K),
    subs(T,Oargs,R).

subs([pn(_H,Lex)|T],Oargs,[Lex|R]) :- !,subs(T,Oargs,R).

subs([H|T],Oargs,[F|R])
 :- member(K,Oargs),
    member(H,K),
    last(F,K),
    subs(T,Oargs,R).

% applicable_term(Form,Term)
% ==========================
% Form => A wff
% Term <= A complex term contained in Form
%
% True if Term is an applicable term contained in Form

% Form is a conjoined wff - don't bother looking

applicable_term(wff(and,_Args),Term) :- \+ Term = pn(_,_),!,fail.
applicable_term(wff(imp,_Args),Term) :- \+ Term = pn(_,_),!,fail.

% Form is an ordinary wff - look down the list of Args
applicable_term(wff(_Pred,Args),Term)
  :-
     applicable_term(Args,Term).

% Form is a list of Args - recurse down it

applicable_term([],[])
  :- !,fail.

applicable_term([H|T],Term)
```

```
  :-
     (applicable_term(H,Term)
      ;
      applicable_term(T,Term)
     ).

applicable_term(Term,Term) :- complex_term(Term).
applicable_term(Term,Term) :- pronominal_term(Term).
applicable_term(Term,Term) :- proper_name_term(Term).


% Form is a complex term - look inside it

applicable_term(term(_,_,Restrict),Term)
  :-
     applicable_term(Restrict,Term).


%=============================================================
%=============================================================
% Low level calls

% terms(Form)
% ===============
% Form => a formula to be analyzed
%
% True if Form contains a complex term.

terms(wff(_Pred,Args))  :- terms(Args).

% Form is a list - recurse down it
terms([H|T]) :- (terms(H) ; terms(T)).

% Form is a complex term - so it contains one.
terms(Form)  :- complex_term(Form).

% subst(Var,Term,Body,Outbody)
% ============================
% Var       => variable to be substituted in
% Term      => term to be replaced by Var
% Body      => input formula to be altered into ..
% Outbody   <= output formula
%
% True if Outbody is the same as Body, but with all occurrences
% of Term replaced by Var.

% no more Body to look at
subst(_,_,[],[]) :- !.

% The body is a wff - look at the args
subst(Var,
      Term,
      wff(Pred,Args),
      wff(Pred,Newargs))
```

```
  :-
      !,
      subst(Var,Term,Args,Newargs).


% The body is a list - recurse down it
subst(Var,
      Term,
      [H_arg|T_args],
      [New_H_arg|New_T_args])
  :-
      !,
      subst(Var,Term,H_arg,New_H_arg),
      !,
      subst(Var,Term,T_args,New_T_args).

% The body is what we're looking for - replace with var.
subst(Var, Term, Term, Var) :- !.

% The body is a complex term - but not the one we want
subst(Var,
      Term,
      term(Quant,NVar,Restrict),
      term(Quant,NVar,Newrestrict))
  :-
      !,
      subst(Var,Term,Restrict,Newrestrict).

% The body is a different atomic case - leave as is.
subst(_,_,Body,Body).

%============================================================
% Real low level calls


precede(Id,Uniq)
 :- name(Id,[_|Idnolist]),
    name(Idno,Idnolist),
    name(Uniq,[_|Uniqnolist]),
    name(Uniqno,Uniqnolist),
    Idno < Uniqno.


complex_term(term(_,_,_)).
pronominal_term(pt(_,_)).
proper_name_term(pn(_,_)).


quantifier(some).
quantifier(every).

nonrefl(he).
nonrefl(him).
nonrefl(she).
```

```
nonrefl(her).
nonrefl(it).
refl(himself).
refl(herself).
refl(itself).

%pretty-print routine

:- op(400,yfx,->).
:- op(400,yfx,&).


pp(every(Vars,Wff),every(Vars,Newout))
 :- !,pp(Wff,Newout).

pp(exists(Vars,Wff),exists(Vars,Newout))
 :- !,pp(Wff,Newout).

pp(wff(imp,[A,B]),(NewA -> NewB))
 :- !,pp(A,NewA),
    pp(B,NewB).

pp(wff(and,[A,B]),(NewA & NewB))
 :- !,pp(A,NewA),
    pp(B,NewB).

pp(wff(Pred,List),Newout)
 :- Newout =.. [Pred|List].
```

A.3.3 Utilities

```
%-----------------------------------------------------
%      Data testing routines
%-----------------------------------------------------

oneoff(S) :- s(S,Syntax),write(Syntax),nl,gen(Syntax,LF),
             pp(LF,PP),
             write('lf:'),
             write(PP),nl,fail.

s(In,Inter) :- init,!,d(Inter,In,[]).


%-----------------------------------------------------
%  counter maintenance
%-----------------------------------------------------

% gvar(Var)
% =========
% generates a new atom

genvar(Var)  :- recorded(index_no,No,_),
                erase_all(index_no),
                New_index is 1 + No,
                recorda(index_no,New_index,_),
                name(x,Inlist),
                name(New_index,Nolist),
                concat(Inlist,Nolist,Newlist),
                name(Var,Newlist).

gnum(New_index)  :- recorded(unique,No,_),
                    erase_all(unique),
                    New_index is 1 + No,
                    recorda(unique,New_index,_).

erase_all(Key) :- \+ erase_aux(Key).

erase_aux(Key) :-  recorded(Key,_,Ref),
                   erase(Ref),
                   !, erase_aux(Key).

% init
% ----
% remove old counters. Set index_no to 0

init :- erase_all(index_no),
        erase_all(unique),
        recorda(index_no,0,_),
        recorda(unique,1,_).

%-----------------------------------------------
% Utility predicates.
%-----------------------------------------------
```

```
member(X,[X|_]).
member(X,[_|T]) :- member(X,T).

last(M,List) :- reverse([M|_],List),!.

concat([],L,L).
concat([H|T1],List,[H|R]) :- concat(T1,List,R).

reverse(Inlist,Outlist)
  :-
     append(Inlist,[Newvar],Diff_in),
     reverse_dlist(Diff_in - Newvar,[Outlist]-[]).

reverse_dlist([A]-A,[Var]-Var).
reverse_dlist([H|L]-Z,Rl-Rz)
  :-
     reverse_dlist(L-Z,Rl-[H|Rz]).

append([],List,List).
append([H|T],List,[H|Rest])
  :-
      append(T,List,Rest).

not(P) :- \+ P.

baggof(A,B,C) :- bagof(A,B,C),!.
baggof(_,_,[]).
```

A.3.4 Test Data Results

```
[john,ran]

lf:ran(john)


[john,saw,mary]

lf:saw(john,mary)


[a,man,ran]

lf:exists([x2],man(x2)&ran(x2))


[every,man,ran]

lf:every([x2],man(x2)->ran(x2))


[a,man,saw,a,woman]

lf:exists([x4,x2],man(x2)&(woman(x4)&saw(x2,x4)))
lf:exists([x2,x4],woman(x4)&(man(x2)&saw(x2,x4)))


[john,saw,every,woman]

lf:every([x3],woman(x3)->saw(john,x3))


[every,man,saw,a,woman]

lf:every([x2],man(x2)->exists([x4],woman(x4)&saw(x2,x4)))
lf:exists([x4],woman(x4)&every([x2],man(x2)->saw(x2,x4)))


[every,man,saw,every,woman]

lf:every([x2],man(x2)->every([x4],woman(x4)->saw(x2,x4)))
lf:every([x4],woman(x4)->every([x2],man(x2)->saw(x2,x4)))


[john,saw,him]

fail


[john,saw,himself]

lf:saw(john,john)
```

```
[he,saw,john]

fail


[himself,saw,john]

fail


[every,man,saw,himself]

lf:every([x2],man(x2)->saw(x2,x2))


[he,saw,every,man]

fail


[a,man,with,a,donkey,ran]

lf:exists([x3,x1],man(x1)&(donkey(x3)&with(x1,x3))&ran(x1))


[every,man,with,a,donkey,saw,a,woman,with,every,pig]

lf:every([x1],
     every([x3],man(x1)&(donkey(x3)&with(x1,x3))
           ->exists([x4],woman(x4)&every([x6],pig(x6)
                                         ->with(x4,x6))&saw(x1,x4))))
lf:exists([x4],woman(x4)&
               every([x6],pig(x6)->with(x4,x6))&
               every([x1],every([x3],man(x1)&(donkey(x3)&with(x1,x3))
                                 ->saw(x1,x4))))


[every,man,with,a,donkey,beat,it]

lf:every([x1],
         every([x3],man(x1)&(donkey(x3)&with(x1,x3))
               ->beat(x1,x3)))


[every,man,with,a,donkey,beat,itself]

lf:every([x1],every([x3],man(x1)&(donkey(x3)&with(x1,x3))->beat(x1,x1)))


[every,man,with,every,donkey,beat,it]

fail
```

```
[every,man,with,john,beat,it]

lf:every([x1],man(x1)&with(x1,john)->beat(x1,john))


[every,man,with,him,ran]

fail


[every,man,with,himself,beat,him]

fail


[every,man,with,himself,beat,himself]

lf:every([x1],man(x1)&with(x1,x1)->beat(x1,x1))


[john,saw,a,donkey,with,him]

lf:exists([x2],donkey(x2)&with(x2,john)&saw(john,x2))


[john,saw,a,donkey,with,himself]

lf:exists([x2],donkey(x2)&with(x2,x2)&saw(john,x2))


[every,man,with,a,donkey,with,him,saw,a,pig,with,him]

lf:every([x1],every([x2],man(x1)&(donkey(x2)&with(x2,x1)&with(x1,x2))
                    ->exists([x4],pig(x4)&with(x4,x2)&saw(x1,x4))))
lf:every([x1],every([x2],man(x1)&(donkey(x2)&with(x2,x1)&with(x1,x2))
                    ->exists([x4],pig(x4)&with(x4,x1)&saw(x1,x4))))


[every,man,with,a,donkey,with,himself,saw,a,pig,with,him]

lf:every([x1],every([x2],man(x1)&(donkey(x2)&with(x2,x2)&with(x1,x2))
                    ->exists([x4],pig(x4)&with(x4,x2)&saw(x1,x4))))
lf:every([x1],every([x2],man(x1)&(donkey(x2)&with(x2,x2)&with(x1,x2))
                    ->exists([x4],pig(x4)&with(x4,x1)&saw(x1,x4))))


[every,man,with,a,donkey,with,a,pig,with,him,beat,him]

lf:every([x1],every([x3,x2],man(x1)&(donkey(x2)&(pig(x3)&
                            with(x3,x2)&with(x2,x3))&with(x1,x2))
                    ->beat(x1,x3)))
lf:every([x1],every([x3,x2],man(x1)&(donkey(x2)&(pig(x3)&
                            with(x3,x2)&with(x2,x3))&with(x1,x2))
```

```
                    ->beat(x1,x2)))
lf:every([x1],every([x3,x2],man(x1)&(donkey(x2)&(pig(x3)&
                            with(x3,x1)&with(x2,x3))&with(x1,x2))
                    ->beat(x1,x3)))
lf:every([x1],every([x3,x2],man(x1)&(donkey(x2)&(pig(x3)&
                            with(x3,x1)&with(x2,x3))&with(x1,x2))
                    ->beat(x1,x2)))


[a,man,ran,he,swam]

lf:exists([x6],man(x6)&ran(x6)&swam(x6))


[a,man,ran,he,swam,he,saw,himself]

lf:exists([x6],man(x6)&ran(x6)&(swam(x6)&saw(x6,x6)))


[every,man,ran,he,swam]

fail


[every,man,saw,john,he,swam]

lf:every([x7],man(x7)->saw(x7,john))&swam(john)


[a,man,saw,every,woman,he,swam]

lf:exists([x9],man(x9)&every([x11],woman(x11)->saw(x9,x11))&swam(x9))


[a,man,saw,a,woman,she,beat,him]

lf:exists([x11,x9],man(x9)&(woman(x11)&saw(x9,x11))&beat(x11,x9))
lf:exists([x11,x9],man(x9)&(woman(x11)&saw(x9,x11))&beat(x9,x11))
lf:exists([x9,x11],woman(x11)&(man(x9)&saw(x9,x11))&beat(x9,x11))
lf:exists([x9,x11],woman(x11)&(man(x9)&saw(x9,x11))&beat(x11,x9))


[a,man,saw,a,woman,she,beat,herself]

lf:exists([x11,x9],man(x9)&(woman(x11)&saw(x9,x11))&beat(x11,x11))
lf:exists([x11,x9],man(x9)&(woman(x11)&saw(x9,x11))&beat(x9,x9))
lf:exists([x9,x11],woman(x11)&(man(x9)&saw(x9,x11))&beat(x9,x9))
lf:exists([x9,x11],woman(x11)&(man(x9)&saw(x9,x11))&beat(x11,x11))


[a,man,with,a,donkey,beat,it,it,saw,him]

lf:exists([x12,x10],man(x10)&(donkey(x12)&with(x10,x12))&
                    beat(x10,x12)&saw(x12,x10))
```

```
lf:exists([x12,x10],man(x10)&(donkey(x12)&with(x10,x12))&
                    beat(x10,x12)&saw(x10,x12))


[a,man,ran,every,donkey,with,him,beat,him]

lf:exists([x6],man(x6)&ran(x6)&
               every([x7],donkey(x7)&with(x7,x6)->beat(x7,x6)))


[if,john,ran,he,swam]

lf:ran(john)->swam(john)


[if,a,man,saw,a,donkey,he,beat,it]

lf:every([x4,x2],man(x2)&(donkey(x4)&saw(x2,x4))->beat(x4,x2))
lf:every([x4,x2],man(x2)&(donkey(x4)&saw(x2,x4))->beat(x2,x4))
lf:every([x2,x4],donkey(x4)&(man(x2)&saw(x2,x4))->beat(x2,x4))
lf:every([x2,x4],donkey(x4)&(man(x2)&saw(x2,x4))->beat(x4,x2))


[if,man,ran,he,swam,he,saw,a,woman]

fail

[if,a,man,ran,he,beat,himself]

lf:every([x2],man(x2)&ran(x2)->beat(x2,x2))
```